(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: **0 342 529 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number: 89108528.4

(22) Date of filing: 11.05.89

(51) Int. Cl.4: G01S 13/02 , G01S 13/87

(30) Priority: 18.05.88 US 195740

(43) Date of publication of application: 23.11.89 Bulletin 89/47

(84) Designated Contracting States: CH DE ES FR GB IT LI SE

(71) Applicant: Hughes Aircraft Company
7200 Hughes Terrace P.O.Box 45066
Los Angeles California 90045-0066(US)

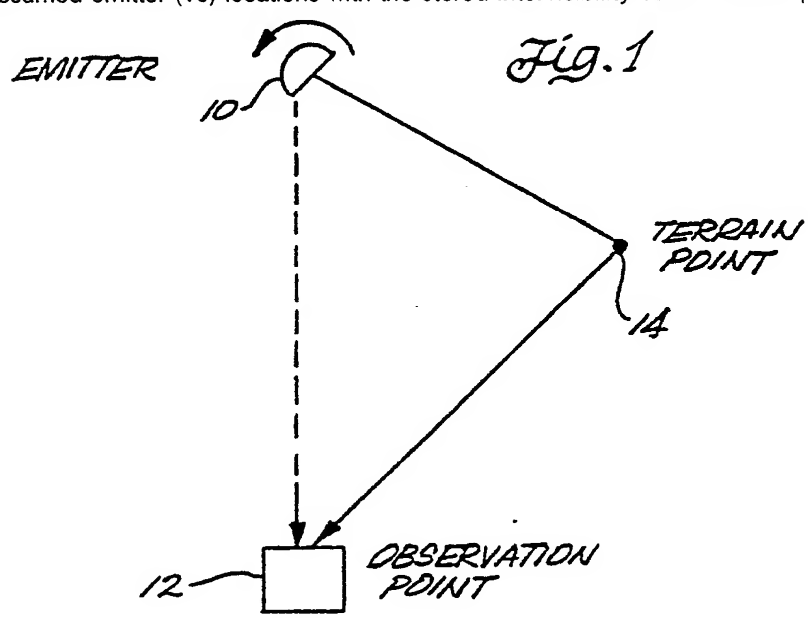
(72) Inventor: Huss, Ronald E.
7800 Airlane Avenue
Los Angeles California 90045(US)
Inventor: Feldmann, Ellis S.
1515 Jefferson Davis Highway
Arlington Virginia 22202(US)

(74) Representative: Witte, Alexander, Dr.-Ing.
Schickhardtstrasse 24
D-7000 Stuttgart(DE)

(54) Method for locating a radio frequency emitter.

(57) A method serves for locating a radio frequency emitter (10) that transmits pulses in a swept beam pattern. The method comprises the steps of storing intervisibility data of terrain points (14) in a region around an observation point, measuring at the observation point (12) the times of arrival of a plurality of terrain point (14) reflections of a single pulse transmitted by the emitter (10), repeating the measuring step for a plurality of pulses transmitted by the emitter (10), and comparing terrain points (14) of reflection calculated from the measured times of arrival for assumed emitter (10) locations with the stored intervisibility data of terrain points (14).

Fig. 1



EP 0 342 529 A2

METHOD FOR LOCATING A RADIO FREQUENCY EMITTER

BACKGROUND OF THE INVENTION

This invention relates to a method for locating a radio frequency emitter that transmits pulses in a swept beam pattern.

In electronic warfare applications, the need arises to locate a radio frequency emitter that transmits pulses in a swept beam. Such a swept beam is usually produced by a rotating antenna, but could also be produced by an oscillating antenna. Current techniques for locating such an emitter require that the observation point lie in the line of sight of the emitter. This requirement means that an emitter can only be located when the observation point is exposed to attack from the emitter. The accuracy of some current techniques for locating a radio frequency emitter also depends upon precise angle measurements, which may be difficult to obtain.

SUMMARY OF THE INVENTION

The invention is a method for locating a radio frequency emitter at an observation point that does not have to be in a direct line of sight from the emitter by using terrain intervisibility data and the relative times of arrival of signals from a single pulse reflected from different points on the terrain at the observation point. The emitter transmits pulses in a regular swept beam pattern. As a result of this regular pattern, the angles of transmission of the pulses can be inferred. Intervisibility data of terrain points in a region around the observation point are stored in computer memory. At the observation point, measurements are made of the times of arrival of a plurality of terrain point reflections of a single pulse transmitted by the emitter. These measurements are repeated for a plurality of pulses transmitted by the emitter. In a computer, a comparison is made of the terrain points of reflection calculated from the measured times of arrival for candidate, i.e., assumed emitter locations with the stored intervisibility data of terrain points. Precise angle measurements are not required to locate a radio frequency emitter in this way.

BRIEF DESCRIPTION OF THE DRAWINGS

The features of a specific embodiment of the best mode contemplated of carrying out the invention are illustrated in the drawings, in which:

FIGS. 1 to 3 are diagrams illustrating spatial considerations used to explain the invention;

FIGS. 4 and 5 are waveforms illustrating time relationships used to explain the invention;

FIG. 6 is a schematic block diagram of apparatus for practicing the invention;

FIG. 7 is a schematic block diagram that illustrates the data used by a computer to locate an emitter in accordance with the principles of the invention; and

FIGS. 8A, 8B, 8C and 8D are diagrams representing the feasibility of various emitter locations.

DETAILED DESCRIPTION OF THE SPECIFIC EMBODIMENT

FIG. 1 is a schematic plan view of a terrain based emitter 10 to be located relative to an observation point 12. It is assumed that emitter 10 rotates at a constant angular velocity of 30° per second and transmits pulsed radio frequency waves, e.g., at 1.344 gigahertz, with a pulse repetition rate, e.g., of 450 pulses per second. It is also assumed that emitter 10 has a directional radiation pattern with a narrow main beam or lobe, e.g., 2° to 3°, and lower intensity side lobes. It is further assumed that the altitude of emitter 10 and observation point 12 through ground reflections and the terrain altitude therebetween is such that observation point 12 is not in a direct line of sight from emitter 10, i.e., observation point 12 is below the line of sight of emitter 10.

Observation point 12 could be a low flying aircraft, a ground site, or a ship on water. When the main beam of emitter 10 is not directed at observation point 12, some of the radio frequency energy from the side lobes reaches observation point 12 through ground reflections in a direct line, as depicted by the broken line in FIG. 1. Some of the radio frequency energy from the main beam also reaches observation

point 12 after lateral reflection from terrain points, such as a point 14, as depicted by the unbroken line in FIG. 1. Thus, each pulse transmitted by emitter 10 reaches observation point 12 in the direct line path and thereafter reaches observation point 12 from a number of laterally reflective paths via various terrain points such as point 14. The time delays between the direct line pulse and the reflected pulses received at observation point 12 are indicative of the specific terrain points from which the delayed pulses are reflected. The longer the transmission path from emitter 10 to the terrain point of reflection and from there to observation point 12, the longer the time delay.

By analyzing the radio frequency energy received at observation point 12 from emitter 10, the angular velocity at which emitter 10 rotates, its pulse repetition rate, and its direction from observation point 12 as a function of time can be determined. Specifically, an extraordinarily large radio frequency energy pulse, hereafter called Peak of Beam (POB), is received at observation point 12 when the main beam of emitter 10 transmits in a direct line to observation point 12. Treating this direct line, i.e., the broken line in FIG. 1, as the angular reference for rotation of emitter 10, the approximate angular position of the main beam of emitter 10 at the time of reception of each direct line pulse at observation point 12 can be inferred. This pulse is, in general, detectable even though the observer does not have direct line of sight to the emitter. Thus, assuming counterclockwise rotation of emitter 10, after 675 pulses from POB, emitter 10 is at an angle of 45° and after 1350 pulses from POB, emitter 10 is at an angle of 90°.

In FIG. 2, point O represents observation point 12 and points E_1 and E_2 represent two emitter locations in the same direction from observation point 12 in a rectangular coordinate system having an I axis and a J axis. The coordinate system is defined so point O is at the origin and points E_1 and E_2 are on the J axis. A given pulse transmitted when the main beam is at an angle θ and arriving at point O after a specified time delay would be reflected from a terrain point F_1 if emitter 10 were located at point E_1 and would be reflected from a terrain point F_2 if emitter 10 were located at point E_2. Thus, for a particular angle θ, and a specified time delay, there is a locus of possible terrain points, represented as a line 16 corresponding to the possible emitter locations. For the particular angle θ and time delays there are different loci of terrain points, shifting downward and to the right in FIG. 2 with increasing time delay.

In FIG. 3, a single emitter location E is assumed. The distance between points E and O, which defines the emitter location relative to observation point 12, is represented by a distance $\underline{r}$. θ is the angle of the main beam at the time of pulse transmission, I is one coordinate of a terrain point of reflection, and J is the other coordinate of the same terrain point of reflection. For a specific location of emitter 10, i.e., point E, and a variable angle θ, the locus of possible terrain points from which a reflected pulse could reach observation point 12, after a given time delay relative to a directly transmitted pulse is defined by an ellipse, as illustrated in FIG. 2, because the reflected transmission paths for all such terrain paths are the same. Thus, the delayed pulses received at observation point 12 correspond to ellipses increasing in size about points O and E with increasing time delay. This relationship is expressed by the equation:

$$\frac{4\,(Y - r/2)^2}{(r + D)^2} + \frac{4X^2}{(r + D)^2 - r^2} = 1 \qquad (1)$$

where the difference between each reflected transmission path, i.e., the sum of the distance from point E to a point (I, J) and the distance from such point (I, J) to point O, and the direct transmission path $\underline{r}$ equals D. The pulse time delay, τ, equals D divided by the speed of light.

Furthermore, since the distance r equals the sum of the distance from point O to point (I, J) J and the distance from point J to point E, the relationship among I, J, $\underline{r}$ and θ can be expressed by the following equation:

$Y = r - X \cot \theta$ (2)

From equations (1) and (2), the coordinates of a point of reflection can be expressed in terms of the distance $\underline{r}$, the angle of the main beam θ, and D, the difference between the reflected and direct transmission paths from point E to point O as follows:

$$X = \frac{(r + D/2)\ \sin\theta}{1 + r/D\ (1 - \cos\theta)} \qquad (3)$$

$$Y = r - \frac{(r + D/2)\ \cos\theta}{1 + r/D\ (1 - \cos\theta)} \qquad (4)$$

The additional information about emitter location that can be obtained from delays due to terrain reflections for successive pulses from the emitter at the assumed pulse repetition rate is not significant. Therefore, only a fraction of the pulses transmitted by the emitter are ordinarily processed in the practice of the invention. By way of example, every 30th pulse transmitted by the emitter could be processed. Thus, for every 2° rotation of θ, a set of time delay data is collected.

FIG. 4 represents the directly transmitted pulses from the emitter received at the observation point. Large pulses 18 represent the POB pulses transmitted at twelve second intervals. Pulses 20 represent the pulses directly transmitted at successive angular positions of the emitter between the POB pulses. For the assumed emitter characteristics, 5,400 pulses 20 appear between successive pulses 18. Each 30th pulse 20 is processed to derive information about the emitter location during a sampling interval T, e.g., 600 microseconds, which is less than the period between pulses 20.

FIG. 5 represents the radio frequency energy from a single pulse received at the observation point from the emitter. Pulse 20, as before, is the directly transmitted pulse. Pulses 22, 24 and 26 are reflections from terrain points in the region around the observation point. A broken horizontal line 28 represents the threshold for discriminating between reflected pulses and noise. The time delay between pulses 20 and 22 is represented as τ_1. The time delay between pulses 20 and 24 is represented as τ_2. The time delay between pulses 20 and 26 is represented as τ_3. Delays τ_1, τ_2, and τ_3 are proportional to the transmission paths from the emitter to the observation points via the terrain points of reflection minus the direct transmission paths from the emitter to the observation point, i.e., $\underline{r}$.

FIG. 6 illustrates apparatus for collecting and processing the pulses from the emitter at the observation point. The radio frequency energy is intercepted by an antenna 30 and fed to a receiver 32, which converts the radio frequency energy to intermediate frequency. A Peak of Beam (POB) detector 34 controls a transmission gate 36. With reference to FIG. 4, detector 34 opens gate 36 for the interval between two successive POB pulses 18, during which a total of 5,400 directly transmitted pulses pass from receiver 32 through gate 36 to a transmission gate 38. These pulses are sensed by a direct pulse detector 40 and applied to a counter 42. After every 30th pulse, counter 42 opens gate 38 for a sampling interval T. The resulting sample as represented in FIG. 5 is coupled to an analog to digital (A/D) converter 44, which digitizes a large number of samples, e.g., 3,000 samples at sampling intervals of 0.2 microsecond. The digitized samples are collected in a buffer storage device 46. After all the samples have been digitized they are transferred en masse to the memory of computer 48.

The emitter is located by comparing the time delays of the reflected pulses with intervisibility data stored in the memory of computer 48. For each terrain point (I, J) in the region around the observation point there are stored in the memory of computer 48 a value of masking depth, $\overline{Z}$ i.e., the height above the terrain point that is visible from the observation point. For a description of a method for determining such intervisibility data, co-pending commonly assigned Application Serial No. 89106258.0, filed on April 8, 1989, by R. E. Huss and R. M. Denlinger (Attorney Docket 2405P171EP), is incorporated fully herein by reference. Computer 48 compares terrain points of reflection (I, J) calculated from the measured times of arrival of a pulse transmitted by the emitter using equations (3) and (4) for candidate, i.e., assumed emitter locations, $\underline{r}$, with the stored intervisibility data of terrain points (I, J). From this comparison, emitter locations

corresponding to some terrain points (I, J) can be eliminated from consideration for the location of the emitter, because of the intervisibility data at such terrain points. For example, the masking depth at a particular terrain point might be so high that a reflection from such terrain point to the observation point would be virtually impossible.

Alternatively, the masking depth at a particular terrain point might be near zero or the terrain point may be visible from the observation point so that a pulse transmitted from an assumed emitter location could have been reflected from that terrain point with the time delay, τ, of the signal received at the observation point; such an assumed emitter location is a good candidate for acceptance as the actual emitter location. By utilizing, in addition, other data about the terrain points such as reflectivity, intervisibility data between the terrain point and the assumed emitter location, and measured time delay data to other observation points, the evaluation of possible emitter locations, vis-a-vis the terrain points in the region around the observation point, can be further refined.

The process is depicted functionally in FIG. 7. Intervisibility data represented by a block 50, namely I, J, and $\bar{Z}^{-}$, and reflected signal data represented by a block 52, namely D and θ are evaluated, as represented by a block 54. The result of this evaluation provides a feasibility of candidate emitter locations at the terrain points in the region about the observation point, as represented by a block 56. As represented by a block 58, other data can also be evaluated to refine the feasibility indication.

FIGS. 8A to 8D represent plots of feasibility of various emitter locations. The feasibility (F) is indicated on the vertical axis, and the terrain points of candidate emitter locations from the observation point (O) are indicated on the J and I axes. The feasibility (F) for each terrain point is determined by counting the number of reflections received at the observation point that could have been transmitted from each terrain point, assuming that it was the emitter location, based on the comparison of time delays of reflected pulses with intervisibility data. The highest value of feasibility (F) occurs at the likely emitter location (E). Thus, FIGS. 8A to 8D depict a scoring function of the possible emitter locations based on the described comparison of the time delays of the reflected pulses with the intervisibility data. Different measures of scoring, i.e., evaluating these comparisons, could be employed to further refine the feasibility data.

Reference is made to Appendix A for a program listing of software for evaluating candidate emitter locations in the described manner on a Digital Equipment Corporation VAX/VMS, Version V4.6 computer.

The described embodiment of the invention is only considered to be preferred and illustrative of the inventive concept; the scope of the invention is not to be restricted to such embodiments. Various and numerous other arrangements may be devised by one skilled in the art without departing from the spirit and scope of this invention.

Claims

1. A method for locating a radio frequency emitter (10) that transmits pulses (18, 20) in a swept beam pattern, characterized by the steps of:
- storing intervisibility data (50) of terrain points (14) in a region around an observation point (12);
- measuring at the observation point (12) the times of arrival (τ) of a plurality of terrain point (14) reflections (52) of a single pulse (18, 20) transmitted by the emitter (10);
- repeating the measuring step for a plurality of pulses (18, 20) transmitted by the emitter (10); and
- comparing terrain points (14) of reflection calculated from the measured times of arrival (τ) for assumed emitter (10) locations with the stored intervisibility data (50) of terrain points (14).

2. The method of claim 1, characterized in that the measuring step comprises measuring the times of arrival (τ) of the plurality of terrain point (14) reflections (52) relative to the time of arrival (τ) of the single pulse (18, 20) directly from the emitter (10).

3. The method of claim 1 or 2, characterized in that the repeating step measures a fraction of the pulses (18, 20) transmitted by the emitter (10).

4. The method of any of claims 1 through 3, characterized by the steps of storing reflectivity data (52) of terrain points (14) in the region and comparing the terrain points (14) of reflection calculated from the measured times of arrival (τ) for assumed emitter (10) locations with the stored relfectivity data (52) of terrain points (14).

APPENDIX A

```
BBBBBBBBBB   7777777777777. .7777777777777777777777777777777   '777   BBBBBBBBBB
BBBBBBBBBB   77777777  Interactive System Design Center  77777777   BBBBBBBBBB
BBBBBBBBBB   777777777777777777777777777777777777777777777777777   BBBBBBBBBB
```

NOTE: JIM

```
  000   RRRR    SSSS  BBBB    000    SSSS
 0   0  R   R  S      B   B  0   0  S
 0   0  R   R  S      B   B  0   0  S
 0   0  RRRR    SSS   BBBB   0   0   SSS
 0   0  R R        S  B   B  0   0      S
 0   0  R  R       S  B   B  0   0      S
  000   R   R  SSSS   BBBB    000   SSSS

PPPP   EEEEE  L        1    N   N  EEEEE  W   W
P   P  E      L       11    N   N  E      W   W
P   P  E      L        1    NN  N  E      W   W
PPPP   EEEE   L        1    N N N  EEEE   W   W
P      E      L        1    N  NN  E      W W W
P      E      L        1    N   N  E      WW WW
P      EEEEE  LLLLL   111   N   N  EEEEE  W   W

       FFFFF   000   RRRR     ;;    333   4  4
       F      0   0  R   R    ;;   3   3  4  4
       F      0   0  R   R             3  4  4
       FFFF   0   0  RRRR     ;;      3   44444
       F      0   0  R R      ;;       3     4
       F      0   0  R  R      ;   3   3     4
       F       000   R   R    ;     333      4
```

File $9SDRB5:[ORSBOS.PEL]PEL1NEW.FOR;34 (800,345,0), last revised on 30-NOV-1987 09:26, is a 43 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 75 bytes.

Job PEL1NEW (187) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:35 from queue SPOCK_LAS2.

```
BBBBBBBBBB   777777777777777777777777777777777777777777777777777   BBBBBBBBBB
BBBBBBBBBB   Digital Equipment Corporation - VAX/VMS Version V4.6   BBBBBBBBBB
BBBBBBBBBB   777777777777777777777777777777777777777777777777777   BBBBBBBBBB
```

```
C
      PROGRAM PEL1NEW
C
C**********************************************************************C
C                                                                      C
C     Program PEL1NEW - Created  5/6/87 from PEL1                      C
C                       Revised  6/1/87, 6/4/87, 6/9/87, 7/6/87        C
C                                                                      C
C     Input Files:                                                     C
C                                                                      C
C       * Terminal                                                     C
C                                                                      C
C     Output Files:                                                    C
C                                                                      C
C       * Terminal                                                     C
C       * PEL1NEW.OUT (Unit = 2)                                       C
C                                                                      C
C     Subroutines Used:                                                C
C                                                                      C
C       * FETCHVIS     - This subroutine reads the two visibility maps. C
C       * DISPVIS      - This subroutine will, if desired, display the C
C                        visibility map on the Ramtek.                 C
C       * SECT         - Determines the boundaries of the "processing  C
C                        sector" to be used. (A square)                C
C       * DISPSECT     - Prints a map showing the "processing sector." C
C                        Does not change any external values.          C
C       * GETTAU       - Gets the returned signal.                     C
C       * GENSUM200    - Processes the returned signal.                C
C       * DISCSET      - Generates the "discriminant function" over the C
C                        map.                                          C
C       * SCOREF       - Calculates the "score" for the Sector.        C
C       * SORT         - Effectively, sorts the sector points into     C
C                        order according to discriminant value.        C
C       * GORF         - Creates table with MD**POWER in descending    C
C                        rank order.                                   C
C       * COUNTSC      - Only writes to Unit 2, does not change any    C
C                        values.                                       C
C       * LINHIST      - Does not change any external values.          C
C       * HISTOC1      - Does not change any external values.          C
C       * HISTOC2      - Does not change any external values.          C
C                                                                      C
C**********************************************************************C
C
        LOGICAL * 4      PFLAG, SDFLAG, HFLAG, RFLAG, BFLAG, CFLAG
        LOGICAL * 4      SECFLG
        CHARACTER        CMD * 1, RDATE * 9, RTIME * 8
        CHARACTER        DIR * 19, TEST_ID * 6
        INTEGER * 2      VISA (0:1118, 0:934)
        INTEGER * 2      INDEX (10000), MAXVAL (10000)
        INTEGER * 2      COORDI (10000), COORDJ (10000), RANK (10000)
        INTEGER * 2      MAPCNT (10000)
        INTEGER * 2      ILIM (-1:1), JLIM (2, 935)
        INTEGER * 2      CRITMD1, CRITMD2
        INTEGER * 4      ITER, KCHAR (4), NCOUNT, MAXK, MMAXK, IRES
        INTEGER * 4      II, K_E, NTAU, MAXTAU, MAXSEC
        INTEGER * 4      IDN, IUP, JDN, JUP, MAXROW, MAXRXC
        INTEGER * 2      IEMROTFLAG, ISCANSTART, ISCANEND
        INTEGER * 2      THRESH, KSTART
        REAL * 4         POWER, DELTA_TAU, RMIN, DISTANCE_FROM_E(10000)
        REAL * 4         PHIO, THETA, RAE, MINDISC, MAXDISC
        REAL * 4         DISTANCE_FROM_B (10000), SUMS (10000)
```

```
      REAL * 4           CU...JM (10000), DISC (10000), DISS   0000)
      REAL * 4           SCORE
      REAL * 4           EATAU (3000), RCRIT, ALPHAR, ALPHAL
      REAL * 4           THETAD, GRAD
C
      INTEGER * 2        BUFFER (-1:3000, 180)    ! BUFFER FOR FILE READ
      INTEGER * 2        NUM_RECORDS              ! # OF BEAM POSITIONS COLLECTED

C
      DATA               MAXTAU /1370/, MAXSEC /10000/
      DATA               GRAD /57.29577951/
      DATA               MAXROW, MAXRXC /935, 1046265/
C
C ******  COMMON blocks:
C
      CHARACTER          BLANKL * 106, TITLE * 106
      INTEGER * 4        JMINP, JMAXP, IMINP, IMAXP, JLEN, ILEN
C
      COMMON /HIST/ BLANKL, TITLE, JMINP, JMAXP, IMINP, IMAXP, JLEN,
     1                ILEN
C
      INTEGER * 4        IO_A, JO_A, IO_A_M, JO_A_M, IO_B, JO_B, IO_B_M
      INTEGER * 4        JO_B_M, IO_E, JO_E, IO_E_M, JO_E_M
C
      COMMON /POSITS/ IO_A, JO_A, IO_A_M, JO_A_M, IO_B, JO_B, IO_B_M,
     1                JO_B_M, IO_E, JO_E, IO_E_M, JO_E_M
C
      INCLUDE 'BUFF.CMN'
C
C ******  Initializations (COMMON block values):
C
      DO II = 1, 106
         BLANKL (II:II) = ' '
      END DO
C
C*********************************************************************C
C
      CALL TIME (RTIME)
      CALL DATE (RDATE)
C
C*********************************************************************C
C                                                                     C
C       Read parameters from operator; Print to output file:         C
C                                                                     C
C       * TITLE      - Header line for output file(s).                C
C       * POWER      - An exponent (power) for masking depth penalty. C
C       * RFLAG      - Indicates if ranking is to be used.            C
C       * CRITMD1    - "Splash points" with masking depths less than  C
C                      (or equal to) this value are treated as        C
C                      "visible", i.e., with masking depth of zero.   C
C       * CRITMD2    - "Splash points" with masking depths greater than C
C                      this value are not processed, i.e., they are   C
C                      treated as absolutely not visible.             C
C       * RMIN       - Minimum range; any point whose range from A is C
C                      less than "rmin" km is given 0 hits.           C
C       * DELTA_TAU  - Minimum delta tau.                             C
C       * RCRIT      - "Critical Radius." Radius of a circle around the C
C                      true emitter location within which the "score" C
C                      does not depend on range.                      C
C       * ALPHAR     - Exponent that determines the strength of range C
C                      dependence of the score, outside of the minimum C
```

```
C                           rang                                       C
C       * ALPHAL       - Scale factor that operates on the Discriminant  C
C                        values in calculating the Score.              C
C       * IDN, IUP     - Limits of the row values in the maps to be read. C
C                        These correspond to the X coordinate.         C
C       * JDN, JUP     - Limits of the column values in the maps to be   C
C                        read. These correspond to the Y coordinate.   C
C       * IEMROTFLAG - Flag for emitter rotation.                      C
C                                 ( 1 = clockwise, 2 = counterclockwise) C
C       * ISCANSTART - Starting scan number.                           C
C       * ISCANEND     - Last scan number.                             C
C       * THRESH       - Threshold for signal strength.                C
C       * KSTART       - Time delay sample starting number.            C
C       * TEST_ID      - Clutter file name.                            C
C       * DIR          - Clutter file directory.                       C
C                                                                      C
C**********************************************************************C
C
        WRITE (*, 1050)
1050    FORMAT ('0','Enter header line for output on next line:')
        READ (*, 1015) TITLE
1015    FORMAT (A80)
C
        WRITE (*, 1055)
1055    FORMAT (' '/'$','Enter Lower masking depth cutoff value: ')
        ACCEPT *, CRITMD1
C
        WRITE (*, 1170)
1170    FORMAT (' '/'$','Enter Higher masking depth cutoff value: ')
        ACCEPT *, CRITMD2
C
        WRITE (*, 1035)
1035    FORMAT (' '/'$','Enter Power (exponent) of masking depth: ')
        ACCEPT *, POWER
C
        WRITE (*, 1060)
1060    FORMAT (' '/'$','Enter minimum range (in Km): ')
        ACCEPT *, RMIN
C
        WRITE (*, 1065)
1065    FORMAT (' '/'$','Enter minimum delta tau (in microseconds): ')
        ACCEPT *, DELTA_TAU
C
        WRITE (*,1125)
1125    FORMAT (' '/'$',
     1  'Enter the critical range for scoring (meters): ')
        ACCEPT *, RCRIT
C
        WRITE (*, 1130)
1130    FORMAT (' '/'$','Enter the range exponent for scoring: ')
        ACCEPT *, ALPHAR
C
        WRITE (*, 1140)
1140    FORMAT (' '/'$','Enter the level scale factor for scoring: ')
        ACCEPT *, ALPHAL
C
        WRITE (*, 1155)
1155    FORMAT (' '/'$','Enter map row limits (IDN,IUP): ')
        ACCEPT *, IDN, IUP
C
        WRITE (*,1160)
```

```
1160    FORMAT (' '/'$','Er  : map column limits (JDN,JUP):
        ACCEPT *, JDN, JUP
C
        IF ((IUP - IDN + 1) .GT. MAXROW) THEN
            WRITE (*, 1157)
            GO TO 9999
        END IF
1157    FORMAT ('0','****  Too many Rows.')
C
        IF ((IUP - IDN + 1) * (JUP - JDN + 1) .GT. MAXRXC) THEN
           WRITE (*,1162)
           GO TO 9999
        END IF
1162    FORMAT ('0','****  Too many Rows times Columns.')
C
C ******  Print parameters to output file (PEL1NEW.OUT):
C
        OPEN (UNIT = 2,FILE = 'PEL1NEW.OUT', STATUS = 'NEW')
C
        WRITE (2, 1020) TITLE
1020    FORMAT ('0', A80)

        WRITE (2, 1105) RDATE, RTIME
1105    FORMAT ('0'/' ',8X,'Program PEL1NEW - Run on ',A9,' at ',A8)
C
        WRITE (2, 1030) CRITMD1
1030    FORMAT ('0'/' ','(M) Lower cutoff value of masking depth = ',I6)
C
        WRITE (2, 1175) CRITMD2
1175    FORMAT ('0','(M) Upper cutoff value of masking depth = ',I6)
C
        WRITE (2, 1025) POWER
1025    FORMAT ('0','(M) Power (exponent) of masking depth = ',F4.1)
C
        WRITE (2, 1051) RMIN
1051    FORMAT ('0','(M) Any point whose range from A is less than ',
     1          F4.1, ' km is given 0 hits')
C
        WRITE (2, 1040) DELTA_TAU
1040    FORMAT ('0','(M) Minimum delta tau = ',F4.2,'  microseconds')

        WRITE (2, 1135) RCRIT, ALPHAR
1135    FORMAT ('0','(M) Range dependence for scoring starts at ',F8.3,
     1          ' meters',
     2      /'0','(M) Exponent for range dependence of score = ',F6.3)
C
        WRITE (2, 1145) ALPHAL
1145    FORMAT ('0','(M) Scale factor for level dependence of score = ',
     1          F6.3)
C
        WRITE (2, 1165) IDN, IUP, JDN, JUP
1165    FORMAT ('0','(M) Map rows are from ',I4,' to ',I4,
     1          /'0','(M) Map columns are from ',I4,' to ',I4)
C
C**********************************************************************C
C                                                                      C
C       Get printing option requests                                   C
C                                                                      C
C**********************************************************************C
C
110     WRITE (*, 1045)
```

```
1045     FORMAT (' '/'$','Do  ou want ranking? (Y or N) ')
         READ (*, 1005) CMD
         IF (CMD .EQ. 'Y' .OR. CMD .EQ. 'y') THEN
               RFLAG=.TRUE.
            ELSE IF (CMD .EQ. 'N' .OR. CMD .EQ. 'n') THEN
               RFLAG=.FALSE.
            ELSE
               WRITE (*,1010)
               GO TO 110
         END IF
C
         IF (.NOT. RFLAG) THEN
            SDFLAG = .FALSE.
            GO TO 193
         END IF
192      WRITE (*, 1092)
1092     FORMAT (' '/'$','Do you want a sector point-by-point detail? ')
         READ (*,1005) CMD
         IF (CMD .EQ. 'Y' .OR. CMD .EQ. 'y') THEN
               SDFLAG=.TRUE.
            ELSE IF (CMD.EQ.'N'.OR.CMD.EQ.'n') THEN
               SDFLAG=.FALSE.
            ELSE
               WRITE (*,1010)
               GO TO 192
         END IF
193      CONTINUE
C
194      WRITE (*, 1094)
1094     FORMAT (' '/'$','Do you want values printed out (table)? ')
         READ (*, 1005) CMD
         IF (CMD .EQ. 'Y' .OR. CMD .EQ. 'y') THEN
               CFLAG=.TRUE.
            ELSE IF (CMD.EQ.'N'.OR.CMD.EQ.'n') THEN
               CFLAG=.FALSE.
            ELSE
               WRITE (*,1010)
               GO TO 194
         END IF
C
195      WRITE (*,1095)
1095     FORMAT (' '/'$','Do you want a linear histogram printed? ')
         READ (*, 1005) CMD
         IF (CMD .EQ. 'Y' .OR. CMD .EQ. 'y') THEN
               BFLAG=.TRUE.
            ELSE IF (CMD .EQ. 'N' .OR. CMD .EQ. 'n') THEN
               BFLAG=.FALSE.
            ELSE
               WRITE (*,1010)
               GO TO 195
         END IF
C
196      WRITE (*, 1096)
1096     FORMAT (' '/'$','Do you want low sensitivity histograms also? ')
         READ (*, 1005) CMD
         IF (CMD .EQ. 'Y' .OR. CMD .EQ. 'y') THEN
               HFLAG=.TRUE.
            ELSE IF (CMD .EQ. 'N' .OR. CMD .EQ. 'n') THEN
               HFLAG=.FALSE.
            ELSE
               WRITE (*,1010)
```

```
            GO TO 196
        END IF
C
197     WRITE (*,1180)
1180    FORMAT (' '/'$','Do you want Full (F) or Restricted area (R) ',
     1          'maps? ')
        READ (*, 1005) CMD
        IF (CMD .EQ. 'F' .OR. CMD .EQ. 'f') THEN
            SECFLG=.TRUE.
          ELSE IF (CMD .EQ. 'R' .OR. CMD .EQ. 'r') THEN
            SECFLG=.FALSE.
          ELSE
            WRITE (*,1010)
            GO TO 197
        END IF
C
C*********************************************************************C
C                                                                     C
C       Get emitter and scan information, threshold for signal        C
C       strength and starting sample number for time delay            C
C                                                                     C
C*********************************************************************C
C
198     TYPE '(A, $)', ' Enter 1 if emitter is rotating clockwise,
     1 -1 if rotating counterclockwise >>> '
        ACCEPT *, IEMROTFLAG
        IF (IEMROTFLAG .NE. 1 .AND. IEMROTFLAG .NE. -1) THEN
           TYPE *, ' BAD VALUE FOR FLAG, RE-ENTER'
           GO TO 198
        END IF
C
        WRITE (2, 1185) IEMROTFLAG
1185    FORMAT ('0','(M) Emitter rotation flag is ',I4,
     1          '     (1 = clockwise, -1 = counterclockwise)')

C
        TYPE '(A, $)', ' Enter first and last scan numbers (1-180) >>> '
        ACCEPT *, ISCANSTART, ISCANEND
C
        WRITE (2, 1205) ISCANSTART, ISCANEND
1205    FORMAT ('0','(M) Starting scan number: ',I4,
     1          /'0','(M) Ending scan number: ',I4)
C
        TYPE '(A, $)', ' Enter the signal strength threshold (1-255) >>> '
        ACCEPT *, THRESH
        WRITE (2, 1206) THRESH
1206    FORMAT ('0','(M) Signal strength threshold: ',I4)
C
        TYPE '(A, $)',' Enter time delay starting sample number (1-3000) >>> '
        ACCEPT *, KSTART
        WRITE (2, 1207) KSTART
1207    FORMAT ('0','(M) Time delay starting sample number: ',I4)
C
C*********************************************************************C
C                                                                     C
C       Get clutter file name and directory.                          C
C                                                                     C
C*********************************************************************C
C
        TYPE '(A, $)', ' ENTER THE CLUTTER FILE NAME >>> '
        ACCEPT 1208, TEST_ID
```

```
1208    FORMAT (A6)
        WRITE (2, 1209) TEST_ID
1209    FORMAT ('0','(M) Clutter file name: ',a6,'.CLUT')
        TYPE '(A, $)', ' ENTER THE CLUTTER FILE DIRECTORY  >>> '
        ACCEPT 1210, DIR
1210    FORMAT (A19)
        WRITE (2, 1211) DIR
1211    FORMAT ('0','(M) Clutter file directory: ',a19)
C
C**********************************************************************C
C                                                                      C
C     Initial calculations:                                            C
C                                                                      C
C**********************************************************************C
C
C ******  Read visibility maps; optionally display:
C
        CALL FETCHVIS (VISA, IDN, IUP, JDN, JUP)
C
        CALL DISPVIS (VISA, IDN, IUP, JDN, JUP)
C
C ******  Set up "low resolution processing sector" map; display:
C
        CALL SECT (RMIN, IO_B, JO_B, DISTANCE_FROM_E, KCHAR, MAXVAL,
     1             COORDI, COORDJ, CUMSUM, IRES, MAPCNT, MAXK,
     2             DISTANCE_FROM_B, MMAXK, NCOUNT, K_E, MAXSEC, ILIM,
     3             JLIM, IDN, IUP, JDN, JUP, SECFLG, ERROR)
C
        CALL DISPSECT (IRES, MAPCNT, IDN, IUP, ILIM, JLIM)
C
C**********************************************************************C
C                                                                      C
C       "Case Loop"                                                    C
C                                                                      C
C       Each iteration of this loop corresponds to a new scan number   C
C       which in turn corresponds to a new value of theta.             C
C                                                                      C
C**********************************************************************C
C
C ******  Initialize Iteration count (# of SCANS processed):

        ITER = 1
C
C ******  DO LOOP stepping through scan numbers:
C
        DO I = ISCANSTART, ISCANEND
C
        IF (ITER .EQ. 1 .OR. ITER .EQ. 2 .OR. ITER .EQ. 4 .OR.
     1      ITER .EQ. 8 .OR. ITER .EQ. 16 .OR. ITER .EQ. 32 .OR.
     2      ITER .EQ. 64 .OR. ITER .EQ. 128 .OR. ITER .EQ. ISCANEND)THEN
           PFLAG = .TRUE.
        ELSE
           PFLAG = .FALSE.
        END IF
C
C ******  Write out scan number being processed
C
        WRITE (*, 1100) ITER
1100    FORMAT (' '/'$','Processing scan # ',I3,': ')
C
C ******  Get TIME DELAY array:
```

```
C
      CALL GETTAU (DELTA_TAU, I, IEMROTFLAG, THRESH, KSTART,
     1                THETAD, MAXTAU, EATAU, NTAU)
C
C ****** Get theta value in radians. Angle was calculated in GETTAU
C ****** to correspond to scan number.
C
      THETA = THETAD / GRAD
C
C ******  Process returned signal:
C
      IF (NTAU .GT. 0) CALL GENSUM200 (NTAU, EATAU, IO_A, JO_A,
     1                                 DISTANCE_FROM_B, POWER,
     2                                 CRITMD1, CRITMD2, THETA, VISA,
     3                                 MAXVAL, CUMSUM, SUMS, IRES,
     4                                 MAPCNT, ILIM, JLIM, IDN, IUP,
     5                                 JDN, JUP)
C
C ******  Print if this is one of the selected Scan values:
C
      IF (.NOT. PFLAG) GO TO 600
C
C ****** Calculate discriminant function at sector points:
C
      CALL DISCSET (MAXK, SUMS, CUMSUM, MAPCNT, DISC, MINDISC,
     1              MAXDISC, DISS)
C
C ******  Calculate the "score":
C
      CALL SCOREF (DISC, MINDISC, MAXDISC, MAPCNT, MAXK, MMAXK, K_E,
     1             DISTANCE_FROM_E, RCRIT, ALPHAR, ALPHAL, SCORE)
C
      IF (RFLAG) CALL SORT (INDEX, NCOUNT, RANK, MAXK, MMAXK, DISC,
     1                      MAPCNT)
C
      IF (CFLAG) CALL COUNTSC (ITER, RANK, DISC, IRES, MAPCNT, RFLAG,
     1                         IDN, IUP, ILIM, JLIM)
C
      IF (BFLAG) CALL LINHIST (ITER, TITLE, MAXK, K_E, MINDISC,
     1                         MAXDISC, DISC, MAPCNT)
C
      IF (SDFLAG) CALL GORF (MAXK, NCOUNT, MMAXK, INDEX, DISC,MAPCNT,
     1                       COORDI, COORDJ, KCHAR, MAXVAL,
     2                       DISTANCE_FROM_E, IRES, POWER, TITLE, K_E,
     3                       0)
C
      IF (HFLAG) CALL HISTOC1 (ITER, MINDISC, MAXDISC, DISC, IRES,
     1                         MAPCNT, SCORE, K_E, IDN, IUP, ILIM,
     2                         JLIM)
C
      CALL HISTOC2 (ITER, RANK, MMAXK, MINDISC, MAXDISC, DISC, IRES,
     1              MAPCNT, RFLAG, SCORE, K_E, IDN, IUP, ILIM, JLIM)
C
C WRITE OUT ESTIMATION OF EMITTER LOCATION AND ERROR
C FROM ACTUAL EMITTER LOCATION.
C
      KK = INDEX (MMAXK + NCOUNT)
C
      PEL_J = FLOAT (COORDJ (KK))
C
      IO = COORDI (KK)
```

```
C
        PEL_I = FLOAT (IO)
C
        WRITE (2, 6100) PEL_I, PEL_J
6100    FORMAT (//,1X, 30H PREDICTED EMITTER LOCATION    ,
     1             2F15.6)
C
        CIMISS = SQRT ( (FLOAT (IO_E) - PEL_I) ** 2 +
     1                  (FLOAT (JO_E) - PEL_J) **2 )
        WRITE (2, 6200)  CIMISS
6200    FORMAT (1X, 30H ERROR IN ESTIMATE              ,
     1             F15.6)
C
C ******  Finished with this SCAN; Get another.
C
600     CONTINUE
        ITER = ITER + 1
C
        END DO
C
C**********************************************************************C
C                                                                      C
C       End of "case loop."                                            C
C                                                                      C
C**********************************************************************C
C
1005    FORMAT (A1)
1010    FORMAT ('O','****  Incorrect entry, try again.')
C
C ******  PROGRAM TERMINATION:
C
        CLOSE (UNIT = 2)
C
9999    WRITE (*, 1110)
1110    FORMAT ('O','                    PEL1NEW bids you adieu',/' ')
C
        STOP
        END
```

```
0000000000    4444444444444  44444444444444444444444444444  4444    0000000000
0000000000    44444444  Interactive System Design Center  44444444    0000000000
0000000000    444444444444444444444444444444444444444444444444444    0000000000

NOTE: JIM

     000    RRRR    SSSS   BBBB    000    SSSS
    0   0   R   R   S      B   B  0   0   S
    0   0   R   R   S      B   B  0   0   S
    0   0   RRRR    SSS    BBBB   0   0   SSS
    0   0   R R        S   B   B  0   0      S
    0   0   R  R       S   B   B  0   0      S
     000    R   R   SSSS   BBBB    000    SSSS

  CCCC    000   U   U  N   N  TTTTT  SSSS   CCCC
 C       0   0  U   U  N   N    T    S      C
 C       0   0  U   U  NN  N    T    S      C
 C       0   0  U   U  N N N    T     SSS   C
 C       0   0  U   U  N  NN    T        S  C
 C       0   0  U   U  N   N    T        S  C
  CCCC    000   UUUUU  N   N    T    SSSS   CCCC

        FFFFF    000    RRRR     ;;     222    666
        F       0   0   R   R    ;;    2   2  6
        F       0   0   R   R             2   6
        FFFF    0   0   RRRR     ;;      2    6666
        F       0   0   R R      ;;     2     6   6
     .. F       0   0   R  R      ;    2      6   6
     .. F        000    R   R    ;     22222   666
```

File _9DRB5:[ORSBOS.PEL]COUNTSC.FOR;26 (322,225,0), last revised on 30-NOV-1987 09:26, is a 16 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 72 bytes.

Job COUNTSC (174) queued to SPOCK_LAS2 on 30-NOV-1987 09:26 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:26 from queue SPOCK_LAS2.

```
0000000000    444444444444444444444444444444444444444444444444444    0000000000
0000000000    Digital Equipment Corporation - VAX/VMS Version V4.6    0000000000
0000000000    444444444444444444444444444444444444444444444444444    0000000000
```

```
      SUBROUTINE COUNTSC (I. .,RANK,DISC,IRES,MAPCNT,RFLAG,I   ,IUP,
     +    ILIM,JLIM)
C**********************************************************************C
C                                                                      C
C     Subroutine COUNTSC - Revision 3/7/86                             C
C                                                                      C
C         This subroutine prints out the count values which correspond C
C     to the histogram points, generated by subroutines HISTOGRAMC1,   C
C     etc. The values printed are somewhat coded, as follows:          C
C                                                                      C
C     For RFLAG = .FALSE.:                                             C
C                                                                      C
C       * '*' - Indicates that the point is not in the "processing     C
C                 sector."                                             C
C       * 'Rmin' - Indicates that the point is inside the "minimum     C
C                  range" from the aircraft. (MAPCNT=-1)               C
C       * 'No H' - Indicates that the point got no "hits." (MAPCNT=0)  C
C       * >>> - Indicates that the value computed was too large to     C
C               print (greater than 99999.5).                          C
C       * <<< - Indicates that the value computed was too large a      C
C               negative value to print (less than -9999.5).           C
C       * A numerical value is just the value of DISC.                 C
C                                                                      C
C     For RFLAG = .TRUE.:                                              C
C                                                                      C
C       * '*' - Indicates that point is not in the "processing         C
C                 sector."                                             C
C       * 'Rmin' - Indicates that the point is inside the "minimum     C
C                  range" from the aircraft. (MAPCNT=-1)               C
C       * 'No H' - Indicates that the point got no "hits." (MAPCNT=0)  C
C       * >>> - Indicates that the value computed was too large to     C
C               print (greater than 99999).                            C
C       * A numerical value is just the value of RANK.                 C
C                                                                      C
C         Note: No external values (COMMON or parameters) are changed  C
C     by this subroutine; only printing is done.                       C
C                                                                      C
C----------------------------------------------------------------------C
C                                                                      C
C     Input Files: None.                                               C
C                                                                      C
C     Output Files:                                                    C
C                                                                      C
C       * Unit 2 - Not opened, from caller.                            C
C                                                                      C
C     Routines Used: None.                                             C
C                                                                      C
C**********************************************************************C
      LOGICAL RFLAG
      CHARACTER*5 CLINE(106)
      INTEGER*2 RANK(10000),PVALS(106),PFLGS(106),MAPCNT(10000),
     +    ILIM(-1:1),JLIM(2,IDN:IUP)
      INTEGER*4 ITER,IRES,IO_R,JO_R,KK,LL,ILEN,IDN,IUP
      REAL*4 DISC(10000),TEMP
C
C ******  COMMON blocks:
C
      CHARACTER BLANKL*106,TITLE*106
      INTEGER*4 JMINP,JMAXP,IMINP,IMAXP,JLEN
      COMMON /HIST/ BLANKL,TITLE,JMINP,JMAXP,IMINP,IMAXP,JLEN
C**********************************************************************C
```

```
C*************************   ******************************   *********C
C
C ******  Page headers:
C
      WRITE (2,1005) TITLE
 1005 FORMAT ('1'/' ',A80)
C
      ILEN = 1 - ( ITER .ge. 10 )
      IF (RFLAG) THEN
                   IF (ITER.eq.1) THEN
                                 WRITE (2,1006) ITER
                               ELSE
                                 WRITE (2,1007) ITER
                               END IF
                 ELSE
                   WRITE (2,1110) ITER
                 END IF
 1006 FORMAT ('0'/'0','Ranking after ',I1,' scan:',/)
 1007 FORMAT ('0'/'0','Ranking after ',I<ILEN>,' scans:',/)
 1110 FORMAT ('0'/'0','Cumulative discriminant after ',I<ILEN>,
     +     ' scans:',/)
C
C ******  Print Histogram like map; first, the column header:
C
      WRITE (2,1035) (LL,LL=JMINP,JMAXP,IRES)
C 1035 FORMAT ('0'/' ',6X,14(X,I4),7(/' ',6X,14(X,I4)))
 1035 FORMAT ('0'/' ',6X,<JLEN>(X,I4),7(/' ',6X,<JLEN>(X,I4)))
C
C ******  Part of map "above" processing sector:
C
      DO 201 LL = 1,JLEN
      CLINE(LL) = '    * '
  201 CONTINUE
C
      DO 205 IO_R = IMINP,ILIM(-1)-1,IRES
      WRITE (2,1030) IO_R, (CLINE(LL),LL=1,JLEN)
  205 CONTINUE
C
C ******  Part of map including processing sector:
C
      KK = 0
      DO 210 IO_R = ILIM(-1), ILIM(1), IRES
C
      DO 200 LL = 1,106
      PFLGS(LL) = -2
  200 CONTINUE
C
      DO 100 JO_R = JLIM(1,IO_R), JLIM(2,IO_R), IRES
      KK = KK + 1
      LL=1+(JO_R-JMINP)/IRES
C
      IF (RFLAG) THEN
                   IF (MAPCNT(KK).LT.0) THEN
                                          PFLGS(LL)=-1
                   ELSE IF (MAPCNT(KK).EQ.0) THEN
                                               PFLGS(LL)=-3
                   ELSE
                     TEMP = RANK(KK)
                     IF (TEMP.GT.99999.5) THEN
                                            PFLGS(LL)=-5
                                          ELSE
```

```
                                                PVALS(LL)=TEMP
                                                PFLGS(LL)=0
                                                GO TO 100
                                              END IF
                          END IF
                        ELSE
                          IF (MAPCNT(KK).LT.0) THEN
                                              PFLGS(LL)=-1
                          ELSE IF (MAPCNT(KK).EQ.0) THEN
                                                   PFLGS(LL)=-3
                          ELSE
                              TEMP = DISC(KK)
                              IF (TEMP.GT.99999.5) THEN
                                                  PFLGS(LL)=-5
                              ELSE IF (TEMP.LT.-9999.5) THEN
                                                       PFLGS(LL)=-6
                              ELSE
                                  PVALS(LL)=NINT(TEMP)
                                  PFLGS(LL)=0
                              END IF
                          END IF
                        END IF
  100 CONTINUE
C
C ******  Construct output line:
C
      DO 120 LL = 1,JLEN
      IF (PFLGS(LL).EQ.0) THEN
          WRITE (CLINE(LL),1020) PVALS(LL)
      ELSE IF (PFLGS(LL).EQ.-2) THEN
          CLINE(LL) = '   * '
      ELSE IF (PFLGS(LL).EQ.-5) THEN
          CLINE(LL) = ' >>> '
      ELSE IF (PFLGS(LL).EQ.-6) THEN
          CLINE(LL) = ' <<< '
      ELSE IF (PFLGS(LL).EQ.-3) THEN
          CLINE(LL) = ' No H'
      ELSE IF (PFLGS(LL).EQ.-1) THEN
          CLINE(LL) = ' Rmin'
      END IF
  120 CONTINUE
 1020 FORMAT (I5)
C
C ******  Print output line:
C
      WRITE (2,1030) IO_R, (CLINE(LL), LL = 1,JLEN)
C 1030 FORMAT (' ',I3,3X,14A5,7(/' ',6X,14A5))
 1030 FORMAT (' ',I3,3X,<JLEN>A5,7(/' ',6X,<JLEN>A5))
  210 CONTINUE
C
C ******  Part of map "below" processing sector:
C
      DO 202 LL = 1,JLEN
      CLINE(LL) = '   * '
  202 CONTINUE
C
      DO 220 IO_R = ILIM(1)+1,IMAXP,IRES
      WRITE (2,1030) IO_R, (CLINE(LL),LL=1,JLEN)
  220 CONTINUE
C
C ******  RETURN TO CALLER:
```

```
C
      RETURN
      END
```

```
PPPPPPPPPP   5555555555555  5555555555555555555555555555555  5555   PPPPPPPPPP
PPPPPPPPPP   55555555  Interactive System Design Center  55555555   PPPPPPPPPP
PPPPPPPPPP   5555555555555555555555555555555555555555555555555555   PPPPPPPPPP
```

NOTE: JIM

```
            000    RRRR     SSSS   BBBB     000     SSSS
           0   0   R   R   S       B   B   0   0   S
           0   0   R   R   S       B   B   0   0   S
           0   0   RRRR     SSS    BBBB    0   0    SSS
           0   0   R R         S   B   B   0   0       S
           0   0   R  R        S   B   B   0   0       S
            000    R   R   SSSS    BBBB     000    SSSS

       DDDD     III    SSSS    CCCC    SSSS   EEEEE  TTTTT
       D   D     I    S       C       S       E        T
       D   D     I    S       C       S       E        T
       D   D     I     SSS    C        SSS    EEEE     T
       D   D     I        S   C           S   E        T
       D   D     I        S   C           S   E        T
       DDDD     III   SSSS     CCCC   SSSS    EEEEE    T

                 FFFFF   000    RRRR     ;;    333
                 F      0   0   R   R    ;;   3   3
                 F      0   0   R   R             3
                 FFFF   0   0   RRRR     ;;      3
                 F      0   0   R R      ;;       3
              .. F      0   0   R  R      ;   3   3
              .. F       000    R   R    ;     333
```

File _9DRB5:[ORSBOS.PEL]DISCSET.FOR;3 (328,255,0), last revised on 30-NOV-1987 09:26, is a 5 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 72 bytes.

Job DISCSET (175) queued to SPOCK_LAS2 on 30-NOV-1987 09:26 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:27 from queue SPOCK_LAS2.

```
PPPPPPPPPP   5555555555555555555555555555555555555555555555555555   PPPPPPPPPP
PPPPPPPPPP   Digital Equipment Corporation - VAX/VMS Version V4.6   PPPPPPPPPP
PPPPPPPPPP   5555555555555555555555555555555555555555555555555555   PPPPPPPPPP
```

```
      SUBROUTINE DISCSET (MAXK,SUMS,CUMSUM,MAPCNT,DISC,MINDISC,MAXDISC,
     +    DISS)
C*********************************************************************C
C                                                                     C
C     Subroutine DISCSET - Revision 2/18/86                           C
C                                                                     C
C                                                                     C
C     Input Files: None.                                              C
C                                                                     C
C     Output Files: None.                                             C
C                                                                     C
C     Routines Used: None.                                            C
C                                                                     C
C*********************************************************************C
      INTEGER*2 MAPCNT(10000)
      INTEGER*4 MAXK,KK
      REAL*4 CUMSUM(10000),DISC(10000),MINDISC,MAXDISC,DISS(10000),
     +    SUMS(10000),MAXR
C
      DATA MAXR/1.7E38/
C*********************************************************************C
C*********************************************************************C
C
C ****** Compute the discriminant fumction, and record min and max
C ****** values:
C
      MAXDISC=-MAXR
      MINDISC=0.0
C
      DO 100 KK=1,MAXK
      IF (MAPCNT(KK).LE.0) THEN
                               DISS(KK)=-MAXR
C
                               DISC(KK)=-MAXR
                             ELSE
                               DISS(KK)=SUMS(KK)/FLOAT(MAPCNT(KK))
C
                               DISC(KK)=CUMSUM(KK)/FLOAT(MAPCNT(KK))
                               MAXDISC=MAX(MAXDISC,DISC(KK))
                               MINDISC=MIN(MINDISC,DISC(KK))
                             END IF
  100 CONTINUE
C
C ****** Assuming that all (valid) DISC values are negative (and thus
C ****** non-zero), calculate a logarithimic discriminant:
C
      DO 200 KK=1,MAXK
      IF (MAPCNT(KK).GT.0) THEN
                               DISC(KK)=-ALOG10(1.0-DISC(KK))
                             END IF
  200 CONTINUE
C
      MINDISC=-ALOG10(1.0-MINDISC)
      MAXDISC=-ALOG10(1.0-MAXDISC)
C
C ****** Return to caller:
C
      RETURN
      END
```

```
QQQQQQQQQQ    6666666666666  66666666666666666666666666666  6666    QQQQQQQQQQ
QQQQQQQQQQ    66666666  Interactive System Design Center  66666666  QQQQQQQQQQ
QQQQQQQQQQ    666666666666666666666666666666666666666666666666666   QQQQQQQQQQ

NOTE: JIM

                    000    RRRR    SSSS   BBBB    000    SSSS
                   0   0   R   R   S      B   B   0   0  S
                   0   0   R   R   S      B   B   0   0  S
                   0   0   RRRR     SSS   BBBB    0   0   SSS
                   0   0   R R         S  B   B   0   0      S
                   0   0   R  R        S  B   B   0   0      S
                    000    R   R   SSSS   BBBB    000    SSSS

             DDDD    III    SSSS   PPPP    SSSS   EEEEE   CCCC   TTTTT
             D   D    I    S       P   P  S       E      C         T
             D   D    I    S       P   P  S       E      C         T
             D   D    I     SSS    PPPP    SSS    EEEE   C         T
             D   D    I        S   P          S   E      C         T
             D   D    I        S   P          S   E      C         T
             DDDD    III   SSSS    P       SSSS   EEEEE   CCCC     T

                           FFFFF    000    RRRR     ;;    666
                           F       0   0   R   R    ;;   6
                           F       0   0   R   R         6
                           FFFF    0   0   RRRR     ;;   6666
                           F       0   0   R R      ;;   6   6
                       ..  F       0   0   R  R      ;   6   6
                       ..  F        000    R   R    ;     666
```

File _9DRB5:[ORSBOS.PEL]DISPSECT.FOR;6 (635,331,0), last revised on 30-NOV-1987 09:26, is a 13 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 73 bytes.

Job DISPSECT (176) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:27 from queue SPOCK_LAS2.

```
QQQQQQQQQQ    666666666666666666666666666666666666666666666666666   QQQQQQQQQQ
QQQQQQQQQQ    Digital Equipment Corporation - VAX/VMS Version V4.6  QQQQQQQQQQ
QQQQQQQQQQ    666666666666666666666666666666666666666666666666666   QQQQQQQQQQ
```

```
      SUBROUTINE dispsect (   .S,MAPCNT,IDN,IUP,ILIM,JLIM)
C***********************************************************************C
C                                                                       C
C     Subroutine SECTDISP - Revision 3/7/86                             C
C                                                                       C
C         This subroutine displays, in the same form as the histogram   C
C     subroutines, a "low resolution" map showing the "processing       C
C     sector" and the aircraft and emitter positions.                   C
C                                                                       C
C     Input Files: None.                                                C
C                                                                       C
C     Output Files: None.                                               C
C                                                                       C
C       * Unit 2 - Not opened, from caller.                             C
C                                                                       C
C     Routines Used: None.                                              C
C                                                                       C
C***********************************************************************C
      CHARACTER OLINE*106,HLINE(106)*1,SYMBOL*1
      INTEGER*2 MAPCNT(10000),ILIM(-1:1),JLIM(2,IDN:IUP)
      INTEGER*4 IO_R,JO_R,KK,LL,IRES,JJ,ILEN,IDN,IUP

      DATA HLINE/106*'+'/
C
C ******  COMMON blocks:
C
      CHARACTER BLANKL*106,TITLE*106
      INTEGER*4 JMINP,JMAXP,IMINP,IMAXP,JLEN
      COMMON /HIST/ BLANKL,TITLE,JMINP,JMAXP,IMINP,IMAXP,JLEN
C
      INTEGER*4 IO_A,JO_A,IO_A_M,JO_A_M,IO_B,JO_B,IO_B_M,JO_B_M,
     +    IO_E,JO_E,IO_E_M,JO_E_M
      COMMON /POSITS/ IO_A,JO_A,IO_A_M,JO_A_M,IO_B,JO_B,IO_B_M,JO_B_M,
     +    IO_E,JO_E,IO_E_M,JO_E_M
C***********************************************************************C
C***********************************************************************C
C
      WRITE (2,1005) TITLE
 1005 FORMAT('1'/' ',A80)
C
C ******  Print legend of symbols:
C
      WRITE (2,1010)
C 1010 FORMAT ('0'/'0',4X,'. indicates a non-sector point;',
C    +            /' ',4X,'* indicates a sector point, outside of the ',
C    +                    'minimum range;',
C    +            /' ',4X,'o indicates a sector point, inside the ',
C    +                    'minimum range.',
C    +           /'0',4X,'A indicates the aircraft, within the sector;',
C    +           /' ',4X,'a indicates the aircraft, outside the sector;',
C    +           /' ',4X,'E indicates the emitter, within the sector;',
C    +           /' ',4X,'e indicates the emitter, outside the sector;',
C    +           /' ',4X,'B or b, if present, indicates the "assumed" ',
C    +    'aircraft position.')
 1010 FORMAT ('0'/'0',4X,'. indicates a non-sector point;',
     +            /' ',4X,'* indicates a sector point, outside of the ',
     +                    'minimum range;',
     +            /' ',4X,'o indicates a sector point, inside the ',
     +                    'minimum range.',
     +           /'0',4X,'E indicates the emitter, within the sector;',
     +           /' ',4X,'e indicates the emitter, outside the sector.')
```

```
C
C ******  Print Histogram:
C
      WRITE (2,1035) JMINP,JMAXP,IRES
 1035 FORMAT ('0'/' ',8X,'J limits are from ',I4,' to ',I4,' by ',I4,
     +      '.'/'0')
C
C ******  If map is too large, start on next page:
C
      ILEN=1+NINT(FLOAT(IMAXP-IMINP)/FLOAT(IRES))
      IF (ILEN.GT.38) WRITE (2,1060)
 1060 FORMAT ('1')
C
C ******  Start of actual "map":
C
      WRITE (2,1025) (HLINE(JJ),JJ=1,JLEN)
 1025 FORMAT (' ',6X,106A1)
C
      KK = 0
      DO 200 IO_R = IMINP, IMAXP, IRES
      OLINE = BLANKL

      LL=0
      DO 300 JO_R = JMINP,JMAXP,IRES
      LL=LL+1
C
      IF (IO_R.GE.ILIM(-1).AND.IO_R.LE.ILIM(1).AND.JO_R.GE.JLIM(1,IO_R)
     +    .AND.JO_R.LE.JLIM(2,IO_R)) THEN
                                     KK=KK+1
                                     IF (MAPCNT(KK).NE.-1) THEN
                                                              SYMBOL='*'
                                                            ELSE
                                                              SYMBOL='o'
                                                            END IF

                                   ELSE
                                     SYMBOL='.'
                                   END IF
C
      IF (IO_R.EQ.IO_E_M.AND.JO_R.EQ.JO_E_M) THEN
                                        IF (SYMBOL.EQ.'.') THEN
                                                              SYMBOL='e'
                                                            ELSE
                                                              SYMBOL='E'
                                                            END IF
                                           END IF
C
C     IF (IO_R.EQ.IO_B_M.AND.JO_R.EQ.JO_B_M) THEN
C                                       IF (SYMBOL.EQ.'.') THEN
C                                                             SYMBOL='b'
C                                                           ELSE
C                                                             SYMBOL='B'
C                                                           END IF
C                                     END IF
C
C     IF (IO_R.EQ.IO_A_M.AND.JO_R.EQ.JO_A_M) THEN
C                                       IF (SYMBOL.EQ.'.') THEN
C                                                             SYMBOL='a'
C                                                           ELSE
C                                                             SYMBOL='A'
C                                                           END IF
C                                     END IF
```

```
C
      OLINE(LL:LL) = SYMBOL
  300 CONTINUE
C
      WRITE (2,1040) IO_R, OLINE(1:JLEN)
 1040 FORMAT (' ',I3,2X,'+',A<JLEN>,'+')
  200 CONTINUE
C
      WRITE (2,1025) (HLINE(JJ),JJ=1,JLEN)
C
C ******  RETURN TO CALLER:
C
      RETURN
      END
```

```
RRRRRRRRRR    7777777777777  77777777777777777777777777777  7777    RRRRRRRRRR
RRRRRRRRRR    77777777  Interactive System Design Center  77777777    RRRRRRRRRR
RRRRRRRRRR    777777777777777777777777777777777777777777777777777    RRRRRRRRRR
```

NOTE: JIM

```
              000    RRRR    SSSS  BBBB    000    SSSS
             0   0  R   R  S      B   B  0   0  S
             0   0  R   R  S      B   B  0   0  S
             0   0  RRRR    SSS   BBBB   0   0   SSS
             0   0  R R        S  B   B  0   0      S
             0   0  R  R       S  B   B  0   0      S
              000   R   R  SSSS   BBBB    000   SSSS

        DDDD     III    SSSS  PPPP   V   V   III    SSSS
        D   D     I    S      P   P  V   V    I    S
        D   D     I    S      P   P  V   V    I    S
        D   D     I     SSS   PPPP   V   V    I     SSS
        D   D     I        S  P      V   V    I        S
        D   D     I        S  P       V V     I        S
        DDDD     III   SSSS   P        V     III   SSSS

                   FFFFF   000   RRRR    ;;   666
                   F      0   0  R   R   ;;  6
                   F      0   0  R   R       6
                   FFFF   0   0  RRRR    ;;  6666
                   F      0   0  R R     ;;  6   6
                   F      0   0  R  R     ;  6   6
                   F       000   R   R   ;    666
```

File _9DRB5:[ORSBOS.PEL]DISPVIS.FOR;6 (654,287,0), last revised on 30-NOV-1987 09:26, is a 4 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 67 bytes.

Job DISPVIS (177) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:27 from queue SPOCK_LAS2.

```
RRRRRRRRRR    777777777777777777777777777777777777777777777777777    RRRRRRRRRR
RRRRRRRRRR    Digital Equipment Corporation - VAX/VMS Version V4.6    RRRRRRRRRR
RRRRRRRRRR    777777777777777777777777777777777777777777777777777    RRRRRRRRRR
```

```
c
      Subroutine dispvis (visa, idn, iup, jdn, jup)
c
      IMPLICIT NONE
c
      integer * 4    xdim /1023/, ydim /1023/
      integer * 4    idn, iup, jdn, jup
      integer * 2    visa (jdn:jup, idn:iup)
      integer * 4    i, j
      integer * 4    xrec, yrec, xem, yem
      character * 1  cmd
c
      byte           image (0:1023, 0:1023)
c
      INTEGER * 4    IO_A, JO_A, IO_A_M, JO_A_M, IO_B, JO_B, IO_B_M
      INTEGER * 4    JO_B_M, IO_E, JO_E, IO_E_M, JO_E_M
c
      COMMON /POSITS/ IO_A, JO_A, IO_A_M, JO_A_M, IO_B, JO_B, IO_B_M,
     1                JO_B_M, IO_E, JO_E, IO_E_M, JO_E_M
c
5     type '(a, $)', ' Do you want the a/c visibility map displayed on
     1 the RAMTEK? (Y or N) >>> '
      read (*, 10) cmd
10    format (a)
      IF (CMD .EQ. 'Y' .OR. CMD .EQ. 'y') THEN
          GO TO 20
        ELSE IF (CMD .EQ. 'N' .OR. CMD .EQ. 'n') THEN
          RETURN
        ELSE
          WRITE (*, 15)
          GO TO 5
      END IF
15    FORMAT ('0','****  Incorrect entry, try again.')
c
20    continue
c
c Find visible points.
c
          do i = 0, min (xdim, iup)
             do j = 0, min (ydim, jup)
                if (visa (j, i) .le. 0) then          ! if unmasked
                      image (j, i) = 1
                end if
                if (visa (j, i) .eq. -1) then         ! Ocean point
                      image (j, i) = 2
                end if
             end do
          end do
c
c Draw crosses at receiver and emitter position.
c
          xrec = IO_A
          yrec = JO_A
          xem  = IO_E
          yem  = JO_E
c
          do i = xrec - 10, xrec + 10
             image (yrec, i) = 3
          end do
          do j = yrec - 10, yrec + 10
             image (j, xrec) = 3
```

```
            end do
c
            do i = xem - 10, xem - 10
               image (yem, i) = 4
            end do
            do j = yem - 10, yem + 10
               image (j, xem) = 4
            end do
c
c Load image to the RAMTEK
c
            call load1024 (0, 0, 1024, 1024, 5, image)
c
        RETURN
        end
```

```
SSSSSSSSSS    8888888888888 88888888888888888888888888888 8888    SSSSSSSSSS
SSSSSSSSSS    88888888  Interactive System Design Center  88888888    SSSSSSSSSS
SSSSSSSSSS    8888888888888888888888888888888888888888888888888    SSSSSSSSSS

NOTE: JIM

                 000    RRRR    SSSS   BBBB    000    SSSS
                0   0   R   R  S       B   B  0   0  S
                0   0   R   R  S       B   B  0   0  S
                0   0   RRRR    SSS    BBBB   0   0   SSS
                0   0   R R        S   B   B  0   0      S
                0   0   R  R       S   B   B  0   0      S
                 000    R   R  SSSS    BBBB    000   SSSS

        FFFFF  EEEEE  TTTTT   CCCC  H   H  V   V   III   SSSS
        F      E        T    C      H   H  V   V    I   S
        F      E        T    C      H   H  V   V    I   S
        FFFF   EEEE     T    C      HHHHH  V   V    I    SSS
        F      E        T    C      H   H  V   V    I       S
        F      E        T    C      H   H   V V     I       S
        F      EEEEE    T     CCCC  H   H    V     III  SSSS

                 FFFFF   000    RRRR     ;;     1     333
                 F      0   0   R   R    ;;    11    3   3
                 F      0   0   R   R           1        3
                 FFFF   0   0   RRRR     ;;     1       3
                 F      0   0   R R      ;;     1        3
              .. F      0   0   R  R      ;     1    3   3
              .. F       000    R   R    ;     111    333
```

File _9DRB5:[ORSBOS.PEL]FETCHVIS.FOR;13 (738,284,0), last revised on 30-NOV-1987 09:26, is a 10 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 76 bytes.

Job FETCHVIS (178) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:27 from queue SPOCK_LAS2.

```
SSSSSSSSSS    8888888888888888888888888888888888888888888888888    SSSSSSSSSS
SSSSSSSSSS    Digital Equipment Corporation - VAX/VMS Version V4.6    SSSSSSSSSS
SSSSSSSSSS    8888888888888888888888888888888888888888888888888    SSSSSSSSSS
```

```
c
      Subroutine Fetchvis (visa, idn, iup, jdn, jup)
c
C********************************************************************C
C                                                                    C
C     Subroutine FETCHVIS - 5/5/87                                   C
C                           Revised 6/4/87                           C
C                                                                    C
C         This subroutine reads in the "visibility files".           C
C         VISA is the high resolution (200m) "aircraft visibility" map.C
C         The aircraft coordinates, JO_A and IO_A, and               C
C         JO_B and IO_B, the emitter coordinates, JO_E and IO_E along C
C         with their respective heights ZAGLA and ZAGLE              C
C         are read in as inputs from the terminal.                   C
C                                                                    C
C--------------------------------------------------------------------C
C                                                                    C
C     Input Files:                                                   C
C                                                                    C
C       * Terminal                                                   C
C       * Unit 1 - VISA data file - Default file name is:            C
C                       IC:[HUSS.GCD]VISAMO.DAT                      C
C                                                                    C
C     Output Files:                                                  C
C                                                                    C
C       * Terminal                                                   C
C       * Unit 2 - Not opened here; received from calling routine(s) C
C                                                                    C
C     Routines Used: None.                                           C
C                                                                    C
C********************************************************************C
C
        character     filea * 40, tstr * 11
        integer * 2   visa (jdn:jup, idn:iup)
        integer * 2   zagla, zagle, md
        integer * 4   jdn, jup, idn, iup
C
C ******  COMMON BLOCKS:
C
        integer * 4 i0_a, j0_a, i0_a_m, j0_a_m, i0_b, j0_b, i0_b_m,
     1              j0_b_m, i0_e, j0_e, i0_e_m, j0_e_m
        common /posits/ i0_a, j0_a, i0_a_m, j0_a_m, i0_b, j0_b, i0_b_m,
     1                  j0_b_m, i0_e, j0_e, i0_e_m, j0_e_m
C********************************************************************C
C
C ******  Read VISA: Visibility map for "true" aircraft position:
C
        type '(a, $)',
     1  ' Enter name of high resolution visibility file: '
        accept 10, FILEA
10      format (a40)
C
        if (filea (1:2) .eq. '  ')
     1     filea = 'ic:[huss.gcd]visamo.dat'
C
        write (2, 20) filea
20      format ('0','(F) visa FILE READ FROM:   ',a40)
C
        open (unit = 1, file = filea, status = 'old',
     1        form = 'unformatted', readonly)
        read (1) visa
```

```
        close (unit = 1)
c
        type '(a, $)',
     1  ' Enter the aircraft (receiver) grid coordinates (x, y) : '
        accept *, xac, yac
        i0_a = nint (xac)
        j0_a = nint (yac)
        type '(a, $)',
     1  ' Enter the aircraft (receiver) height AGL in meters : '
        accept *, htac
        zagla = nint (htac)
c
c ***** IO_B, JO_B, AND ZAGLB are just SECOND SET of variables where the
c ***** a/c coordinates and height agl are stored.
c
        i0_b = i0_a
        j0_b = j0_a
        zaglb = zagla
C
        write (2, 40) i0_a, j0_a, zagla
40      format ('0','(F) ',4X,'Aircraft  (A)  position = (',I4,',',I4,
     1          ',',I6,')')
C
C ******  Get emitter's position and height AGL
C
        type '(a, $)',
     1  ' Enter the emitter grid coordinates (x, y) : '
        accept *, xem, yem
        i0_e = nint (xem)
        j0_e = nint (yem)
        type '(a, $)',
     1  ' Enter the emitter height AGL in meters : '
        accept *, htem
        zagle = nint (htem)
c
        write (2, 50) i0_e, j0_e, zagle
50      format
     1  ('0','(F) ',4X,'Emitter  (E)  position = (',I4,',',I4,',',
     2          I6,')')
C
C ******  Print visibility:
C
        md = visa (j0_e, i0_e)
        if (zagle .le. md) then
            tstr = '(Invisible)'
          else
            tstr = '(Visible)  '
        end if
c
        write (2, 60) md, tstr
60      format ('0','(F) Masking depth at E position from A = ',I5,
     1    ' meters',2X,A11)
C
C ******  Return to caller:
C
        return
        end
```

```
TTTTTTTTTT     9999999999999  99999999999999999999999999999999  9999     TTTTTTTTTT
TTTTTTTTTT     99999999  Interactive System Design Center  99999999     TTTTTTTTTT
TTTTTTTTTT     9999999999999999999999999999999999999999999999999999     TTTTTTTTTT
```

NOTE: JIM

```
            000     RRRR     SSSS   BBBB     000     SSSS
           0   0    R   R   S       B   B   0   0   S
           0   0    R   R   S       B   B   0   0   S
           0   0    RRRR     SSS    BBBB    0   0    SSS
           0   0    R R         S   B   B   0   0       S
           0   0    R  R        S   B   B   0   0       S
            000     R   R   SSSS    BBBB     000    SSSS

  FFFFF   III   L       EEEEE   RRRR    EEEEE    AAA    DDDD
  F        I    L       E       R   R   E       A   A   D   D
  F        I    L       E       R   R   E       A   A   D   D
  FFFF     I    L       EEEE    RRRR    EEEE    A   A   D   D
  F        I    L       E       R R     E       AAAAA   D   D
  F        I    L       E       R  R    E       A   A   D   D
  F       III   LLLLL   EEEEE   R   R   EEEEE   A   A   DDDD

                FFFFF    000    RRRR      ;;     1
                F       0   0   R   R     ;;    11
                F       0   0   R   R            1
                FFFF    0   0   RRRR      ;;     1
                F       0   0   R R       ;;     1
            ..  F       0   0   R  R       ;     1
            ..  F        000    R   R     ;     111
```

File _9DRB5:[ORSBOS.PEL]FILEREAD.FOR;1 (769,271,0), last revised on 30-NOV-1987 09:26, is a 3 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 68 bytes.

Job FILEREAD (179) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:27 from queue SPOCK_LAS2.

```
TTTTTTTTTT     9999999999999999999999999999999999999999999999999999     TTTTTTTTTT
TTTTTTTTTT     Digital Equipment Corporation - VAX/VMS Version V4.6     TTTTTTTTTT
TTTTTTTTTT     9999999999999999999999999999999999999999999999999999     TTTTTTTTTT
```

```
      subroutine FILEREAD  .IUM_RECORDS, RECORD_SIZE, FILE_   'IE, BUFFER)
      implicit none

      integer * 2 M
      parameter (M = 180)      ! second dimension of returned buffer
c
c;    WHERE:
c;
c;       INTEGER*2    NUM_RECORDS     # OF RECORDS TO BE READ FROM THE FILE
c;       INTEGER*2    RECORD_SIZE     SIZE OF EACH RECORD IN BYTES
c;       CHARACTER*N  FILE_NAME       NAME OF FILE TO BE OPENED AND READ
c;       INTEGER*2    BUFFER(RECORD_SIZE,M)
c;                                    THE BUFFER TO RECEIVE THE RECORDS
c;                                    READ FROM THE FILE. M>=NUM_RECORDS.
c;                                    NOTE: BUFFER MUST HAVE EXACTLY
c;                                          RECORD_SIZE ROWS.
      INTEGER * 2          NUM_RECORDS          !# OF RECORDS TO BE READ FROM THE FILE
      INTEGER * 2          RECORD_SIZE          ! SIZE OF EACH RECORD IN BYTES
      CHARACTER * (*)      FILE_NAME            ! NAME OF FILE TO BE OPENED AND READ
      INTEGER * 2          BUFFER (RECORD_SIZE / 2, M)
      integer * 2          k, rc_ov_2

      open (unit = 99, file = file_name, form = 'unformatted',
     1        status = 'old', readonly)

      rc_ov_2 = record_size / 2
      num_records = 1

      do while (.true.)

        read (99, end = 999) (buffer (k, num_records), k = 1, rc_ov_2)

        num_records = num_records + 1

      end do

999   continue

      close (99)

      num_records = num_records - 1

      return
      end
```

```
UUUUUUUUUU   0000000000000 00000000000000000000000000000000 0000   UUUUUUUUUU
UUUUUUUUUU   00000000  Interactive System Design Center  00000000   UUUUUUUUUU
UUUUUUUUUU   00000000000000000000000000000000000000000000000000   UUUUUUUUUU
```

NOTE: JIM

```
          000    RRRR    SSSS   BBBB    000    SSSS
         0   0   R   R  S       B   B  0   0  S
         0   0   R   R  S       B   B  0   0  S
         0   0   RRRR    SSS    BBBB   0   0   SSS
         0   0   R R        S   B   B  0   0      S
         0   0   R  R       S   B   B  0   0      S
          000    R   R  SSSS    BBBB    000   SSSS

 GGGG  EEEEE  N   N  SSSS  U   U  M   M   222    000    000
G      E      N   N S      U   U  MM MM  2   2  0   0  0   0
G      E      NN  N S      U   U  M M M      2  0  00  0  00
G      EEEE   N N N  SSS   U   U  M   M     2   0 0 0  0 0 0
G GGG  E      N  NN     S  U   U  M   M    2    00  0  00  0
G   G  E      N   N     S  U   U  M   M   2     0   0  0   0
 GGG   EEEEE  N   N SSSS    UUUUU M   M  22222   000    000

             FFFFF   000   RRRR    ;;   4  4
             F      0   0  R   R   ;;   4  4
             F      0   0  R   R        4  4
             FFFF   0   0  RRRR    ;;   44444
             F      0   0  R R     ;;      4
             F      0   0  R  R     ;      4
             F       000   R   R    ;      4
```

File _9DRB5:[ORSBOS.PEL]GENSUM200.FOR;4 (779,244,0), last revised on 30-NOV-1987 09:26, is a 20 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 80 bytes.

Job GENSUM200 (180) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:27 from queue SPOCK_LAS2.

```
UUUUUUUUUU   00000000000000000000000000000000000000000000000000   UUUUUUUUUU
UUUUUUUUUU   Digital Equipment Corporation - VAX/VMS Version V4.6   UUUUUUUUUU
UUUUUUUUUU   00000000000000000000000000000000000000000000000000   UUUUUUUUUU
```

```
      SUBROUTINE GENSUM200 ( ITAU,EATAU,IO_A,JO_A,DISTANCE_FROM_B,POWER,
     +     CRITMD1,CRITMD2,THETA,VISAB,MAXVAL,CUMSUM,SUMS,IRES,MAPCNT,
     +     ILIM,JLIM,IDN,IUP,JDN,JUP)
C**********************************************************************C
C                                                                      C
C     Subroutine GENSUM200 - Revision 3/12/86, 7/6/87                  C
C                                                                      C
C         This subroutine processes the  returned signal which is      C
C     read in by GETTAU. This routine is called once for               C
C     each value of THETA, that is, for each transmitted pulse. The    C
C     results are sums of values which can be further processed when   C
C     final results are desired.                                       C
C         Basically, each point in the "sector," and outside of the    C
C     minimum range of the (assumed) aircraft position, is looked at   C
C     in turn as a candidate emitter location. For each point the      C
C     processing is as follows.                                        C
C         At the start of processing a pulse, a sum and a counter are  C
C     both set to zero. Then, for each time when there is a received   C
C     signal present, there will be a map point (not necessarily a     C
C     sector point) which would be the corresponding scattering point. C
C     If the indicated point is outside of the data base, we ignore    C
C     it.                                                              C
C         If however, the point is inside of our map, we increment the C
C     counter for this candidate emitter location and add this         C
C     scattering points contribution to the sum for this candidate     C
C     emitter. Whether the candidate emitter location is a good one    C
C     depends on the visibility of this scattering point from the      C
C     aircraft's location. The value added is related to the masking   C
C     depth of the scattering point. It is the negative of the masking C
C     depth taken to some power: -(MD**POWER). Actually, only masking  C
C     depths which are greater (i.e., worse) than some selected value  C
C     are added to the sum; if they are below (better) than this       C
C     value, they are essentially set to zero.                         C
C         Finally, for each candidate emitter point, the above sum,    C
C     which is just for this "pulse," is added to an overall sum which C
C     is the sum for the entire series of pulses. And a total counter  C
C     is kept.                                                         C
C                                                                      C
C----------------------------------------------------------------------C
C                                                                      C
C     Input Files: None.                                               C
C                                                                      C
C     Output Files: None.                                              C
C                                                                      C
C     Routines Used: None.                                             C
C                                                                      C
C**********************************************************************C
      INTEGER*2 VISAB(JDN:JUP,IDN:IUP),MAXVAL(10000),MD,
     +     MAPCNT(10000),ILIM(-1:1),JLIM(2,IDN:IUP),CRITMD1,CRITMD2
      INTEGER*4 ITAU,IO_R,JO_R,KK,IP,JP,IRES,NTAU,IO_A,JO_A,JDN,JUP,IDN,
     +     IUP
      REAL*4 POWER,ONE_MINUS_COSTHETA,THETA,BETA,DELX,DELY,
     +     SINBETA,COSBETA,PI,RNORM,DIST,P1,CUMSUM(10000),SUMS(10000),
     +     DISTANCE_FROM_B(10000),EATAU(NTAU)

        INTEGER * 4  ILL, JLL, J, I
        REAL * 4     XSP, YSP
        INTEGER * 2  MD_PREV, COUNT
C
      PARAMETER ( PI = 3.14159 )
C**********************************************************************C
```

```
C*************************** ****************************** ********C
C
      ONE_MINUS_COSTHETA = 1.0 - COS(THETA)
C
C ******  Check each sector point, within resolution IRES. Candidate
C ******  emitter is located at map coordinates (JO_R,IO_R). Locations
C ******  are serially indexed by KK:
C
      KK = 0
      DO 200 IO_R = ILIM(-1), ILIM(1), IRES
      DO 210 JO_R = JLIM(1,IO_R), JLIM(2,IO_R), IRES
      KK = KK + 1
C
C ******  If this point lies within the minimum range of A, hits are
C ******  not counted; this is indicated by MAPCNT(KK) to -1; go to
C ******  next point:
C
      IF (MAPCNT(KK).EQ.-1) GO TO 210
C
C ******  Calculate "absolute" transmission angle from candidate
C ******  emitter:
C
      DELX=FLOAT(IO_R-IO_A)
      DELY=FLOAT(JO_R-JO_A)
      BETA = PI + THETA + ATAN2 (DELY, DELX)
      SINBETA = SIN(BETA)
      COSBETA = COS(BETA)
C
C ******  For each tau value:
C
      SUMS(KK) = 0.0
      DO 100 ITAU = 1,NTAU
C
C ******  compute location of splash point in map coordinates
C
      DIST = 0.3*EATAU(ITAU)
      RNORM = DISTANCE_FROM_B(KK)/DIST
      P1 = DIST*(RNORM + 0.5) / (1.0 + RNORM*ONE_MINUS_COSTHETA)
C
      JP = JO_R + nint ( 5.0 * P1 * SINBETA )
      IP = IO_R + nint ( 5.0 * P1 * COSBETA )
C
      YSP = FLOAT (JO_R) + ( 5.0 * P1 * SINBETA )
      XSP = FLOAT (IO_R) + ( 5.0 * P1 * COSBETA )
      JLL = INT (YSP)
      ILL = INT (XSP)
C
C ******  If splash point is out of out map's boundaries, ignore it and
C ******  go on and check next TAU value:
C
      IF (JP.LT.JDN .OR. JP.GT.JUP .OR. IP.LT.IDN
     +    .OR. IP.GT.IUP) GO TO 100
C
C ******  Splash point lies within our map's boundaries; MD is masking
C ******  depth at splash point:
C
C
C ******  Since the radar beam is not a straight line but has some width
C ******  a splash point could be any point within a strip in this beam.
C ******  A crude way to simulate this is to look for the lowest masking
C ******  depth within a box around the splash point. The dimensions of the
```

```
C ******\ vary depending ho\  ar out from the candidate emit    we are
C ******  i.e. how large the time delay is.
C
      IF (EATAU (ITAU) .LE. 111.0) THEN
          MD_PREV = 10000
          COUNT = 0
          DO I = MAX (ILL, IDN), MIN (ILL + 1, IUP)      ! Process points in
             DO J = MAX (JLL, JDN), MIN (JLL + 1, JUP)  ! 2x2 box around point
                IF (VISAB (J, I) .NE. -1) THEN
                    MD = MIN (VISAB (J, I), MD_PREV)
                  ELSE
                    COUNT = COUNT + 1
                END IF
                MD_PREV = MD
             END DO
          END DO
          IF (COUNT .GT. 2) MD = -1        ! if sea points more than half md=-1
        ELSE IF (EATAU (ITAU).GT.111.0 .AND. EATAU (ITAU).LE.253.0)THEN
          MD_PREV = 10000
          COUNT = 0
          DO I = MAX (ILL - 1, IDN), MIN (ILL + 2, IUP)     ! Process points in
             DO J = MAX (JLL - 1, JDN), MIN (JLL + 2, JUP) ! 4x4 box
                IF (VISAB (J, I) .NE. -1) THEN
                    MD = MIN (VISAB (J, I), MD_PREV)
                  ELSE
                    COUNT = COUNT + 1
                END IF
                MD_PREV = MD
             END DO
          END DO
          IF (COUNT .GT. 8) MD = -1        ! if sea points more than half md=-1
        ELSE IF (EATAU (ITAU).GT.253.0 .AND. EATAU (ITAU).LE.401.0)THEN
          MD_PREV = 10000
          COUNT = 0
          DO I = MAX (ILL - 2, IDN), MIN (ILL + 3, IUP)     ! Process points in
             DO J = MAX (JLL - 2, JDN), MAX (JLL + 3, JUP) ! 6x6 box
                IF (VISAB (J, I) .NE. -1) THEN
                    MD = MIN (VISAB (J, I), MD_PREV)
                  ELSE
                    COUNT = COUNT + 1
                END IF
                MD_PREV = MD
             END DO
          END DO
          IF (COUNT .GT. 18) MD = -1       ! if sea points more than half md=-1
        ELSE
          MD_PREV = 10000
          COUNT = 0
          DO I = MAX (ILL - 3, IDN), MIN (ILL + 4, IUP)     ! Process points in
             DO J = MAX (JLL - 3, JDN), MAX (JLL + 4, JUP) ! 8x8 box
                IF (VISAB (J, I) .NE. -1) THEN
                    MD = MIN (VISAB (J, I), MD_PREV)
                  ELSE
                    COUNT = COUNT + 1
                END IF
                MD_PREV = MD
             END DO
          END DO
          IF (COUNT .GT. 32) MD = -1       ! if sea points more than half md=-1
      END IF
C
```

```
C IF SPLASH POINT IS A SEA . .NT (I. E. MD = -1) DO NOT PROC. .5.
C
        IF (MD .EQ. -1) GO TO 100
C
C ******  MD's falling between minus infinite and CRITMD1 are treated
C ******  as visible (i.e. masking depth of zero). Those between CRITMD1
C ******  and CRITMD2 are included in the Discriminant. Those above
C ******  CRITMD2 are treated as completely invisible, i.e., not
C ******  processed at all:
C
      IF (MD.LE.CRITMD1) THEN
                                MAPCNT(KK)=MAPCNT(KK)+1
      ELSE IF (MD.LE.CRITMD2) THEN
                                   MAPCNT(KK)=MAPCNT(KK)+1
                                   SUMS(KK) = SUMS(KK) - FLOAT(MD)**POWER
                                   MAXVAL(KK) = MAX ( MAXVAL(KK), MD )
      END IF
C
  100 CONTINUE
C
C ******  All TAU values processed; summary for this candidate emitter
C ******  location (segment point #KK):
C
      IF (MAPCNT(KK).GT.0) THEN
                                  CUMSUM(KK) = CUMSUM(KK) + SUMS(KK)
                                END IF
  210 CONTINUE
  200 CONTINUE
C
C ******  RETURN TO CALLER:
C
      RETURN
      END
```

```
VVVVVVVVVV    1111111111111. 11111111111111111111111111111111   111    VVVVVVVVVV
VVVVVVVVVV    11111111  Interactive System Design Center  11111111    VVVVVVVVVV
VVVVVVVVVV    111111111111111111111111111111111111111111111111111    VVVVVVVVVV
```

NOTE: JIM

```
  000    RRRR     SSSS  BBBB    000     SSSS
 0   0   R   R   S      B   B  0   0   S
 0   0   R   R   S      B   B  0   0   S
 0   0   RRRR     SSS   BBBB   0   0    SSS
 0   0   R R         S  B   B  0   0       S
 0   0   R  R        S  B   B  0   0       S
  000    R   R   SSSS   BBBB    000    SSSS

  GGGG   EEEEE  TTTTT  DDDD    AAA   TTTTT
 G       E        T    D   D  A   A    T
 G       E        T    D   D  A   A    T
 G       EEEE     T    D   D  A   A    T
 G GGG   E        T    D   D  AAAAA    T
 G   G   E        T    D   D  A   A    T
  GGG    EEEEE    T    DDDD   A   A    T

  FFFFF   000   RRRR      ;;    222   55555
  F      0   0  R   R     ;;   2   2  5
  F      0   0  R   R              2  555
  FFFF   0   0  RRRR      ;;      2      5
  F      0   0  R R       ;;     2       5
..F      0   0  R  R       ;    2     5  5
..F       000   R   R      ;   22222   555
```

File _9DRB5:[ORSBOS.PEL]GETDAT.FOR;25 (781,278,0), last revised on 30-NOV-1987 09:26, is a 4 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 74 bytes.

Job GETDAT (181) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:28 from queue SPOCK_LAS2.

```
VVVVVVVVVV    111111111111111111111111111111111111111111111111111    VVVVVVVVVV
VVVVVVVVVV    Digital Equipment Corporation - VAX/VMS Version V4.6    VVVVVVVVVV
VVVVVVVVVV    111111111111111111111111111111111111111111111111111    VVVVVVVVVV
```

```
c  Subroutine getdat.for --  etches the scan data of run '0C  '01' from
c  the disk and puts it in BUFFER(-1:3000,180)

        subroutine getdat

        IMPLICIT        NONE

        INTEGER*2       BUFFER(-1:3000,180),     ! BUFFER FOR FILE READ
     *                  RECORD_SIZE,             ! RECORD SIZE IN BYTES
     *                  NUM_RECORDS,             ! # OF BEAM POSITIONS COLLECTED
     *                  spec_len

        CHARACTER       DATE_TIME*23,            ! DATE AND TIME
     *                  FILE_NAME*72,            ! NAME OF GENERIC DATA FILE TO READ
     *                  DIR*19,                  ! DIRECTORY NAME OF FILE
     *                  TEST_ID*6                ! TEST ID # TO READ

        include '[orsrmb.pellnew]buff.cmn'

        data RECORD_SIZE /6004/  ! = 2 * NUM_SAMPLES + 4, NUM_SAMPLES = 3000
        data spec_len /30/

c  SPEC_LEN = # OF CHARS IN FILE SPEC

C  FORM THE FILE SPECIFICATION

        FILE_NAME(1:SPEC_LEN) = DIR//TEST_ID//'.CLUT'

C  READ THE FILE - RECORD SIZE AND FILE NAME ARE INPUTS TO FILEREAD,WHILE
C  NUM RECORDS IS RETURNED ALONG WITH THE DATA IN BUFFER

        CALL FILEREAD(NUM_RECORDS,       ! # OF AZIMUTH BEAM POSITIONS (returned)
     *                RECORD_SIZE,       ! SIZE OF EACH RECORD (passed)
     *                FILE_NAME,         ! FILE SPEC OF FILE TO BE READ (passed)
     *                BUFFER)            ! BUFFER TO RECEIVE THE DATA (returned)

c  BUFFER(-1:3000,180) contains the complete file of scan data.
c                      RECORD # is in BUFFER(-1, i ) and the number
c                      of samples is in BUFFER(0,i).

        WRITE(6,40) FILE_NAME(1:SPEC_LEN)

 40     FORMAT(1X,'FINISHED READING FILE',1X,A<SPEC_LEN>)
        WRITE(6,45) NUM_RECORDS
 45     FORMAT(1X,'DATA COLLECTED ON ',I4,' BEAM POSITIONS.',/)

        return

        END
```

```
WWWWWWWWWW    2222222222222.  .2222222222222222222222222222.  222    WWWWWWWWWW
WWWWWWWWWW    22222222  Interactive System Design Center  22222222    WWWWWWWWWW
WWWWWWWWWW    222222222222222222222222222222222222222222222222    WWWWWWWWWW
```

NOTE: JIM

```
 000    RRRR    SSSS   BBBB    000    SSSS
0   0   R   R  S       B   B  0   0  S
0   0   R   R  S       B   B  0   0  S
0   0   RRRR    SSS    BBBB   0   0   SSS
0   0   R R        S   B   B  0   0      S
0   0   R  R       S   B   B  0   0      S
 000    R   R  SSSS    BBBB    000   SSSS


 GGGG   EEEEE   TTTTT   TTTTT    AAA    U   U
G       E         T       T     A   A   U   U
G       E         T       T     A   A   U   U
G       EEEE      T       T     A   A   U   U
G GGG   E         T       T     AAAAA   U   U
G   G   E         T       T     A   A   U   U
 GGG    EEEEE     T       T     A   A   UUUUU

FFFFF    000    RRRR      ;;     333    55555
F       0   0   R   R     ;;    3   3   5
F       0   0   R   R               3   555
FFFF    0   0   RRRR      ;;      3        5
F       0   0   R R       ;;        3       5
F       0   0   R  R       ;    3   3   5   5
F        000    R   R      ;     333     555
```

File 9DRB5:[ORSBOS.PEL]GETTAU.FOR;35 (786,368,0), last revised on 30-NOV-1987 09:26, is a 5 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 74 bytes.

Job GETTAU (182) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:28 from queue SPOCK_LAS2.

```
WWWWWWWWWW    222222222222222222222222222222222222222222222222    WWWWWWWWWW
WWWWWWWWWW    Digital Equipment Corporation - VAX/VMS Version V4.6    WWWWWWWWWW
WWWWWWWWWW    222222222222222222222222222222222222222222222222    WWWWWWWWWW
```

```
c
      Subroutine Gettau (delta_tau, iscan, iemrotflag, thresh, kstart,
     1                   thetad, maxtau, eatau, ntau)
c
c SUBROUTINE GETTAU May-6-1987, REVISED JUNE-9-1987, REVISED SEP-21-1987
c
c This subroutine reads in the real data (which is stored in the array
c buffer) and fills the output array eatau which contains time delays
c of splash points.
c
c
      integer * 2    buffer(-1:3000, 180)        ! buffer for file read
      integer * 2    num_records, num_samples
c
      include 'buff.cmn'
c
      integer * 4    maxtau, ntau
      real * 4       eatau (maxtau)
      real * 4       delta_tau, thetad, oldtau, tauv
      integer * 2    iscan, iemrotflag
      integer * 2    thresh
      integer * 2    kstart
      data           num_samples /3000/
c
c Fetch the data from the storage file.
c
      call getdat
c
c Now all of the scan data are in BUFFER (i,j), j=scan no., i=sample no.
c Go through the data to get the splash points.
c
c Initialize time delay counter and time delay store variable
c
      ntau = 0
      oldtau = 0.0
c
c  Check that iscan (scan no.) is between 1 and 180. If not adjust.
c
      iscan = MIN (iscan, 180)
      iscan = MAX (iscan, 1)

      if (iemrotflag .eq. 1) then
          thetad = 360.0 -
     1              ( (float (iscan - 1) / (float (num_records) - 0.5))
     2                * 360.0 )
        else if (iemrotflag .eq. -1) then
          thetad =
     1              ( (float (iscan - 1) / (float (num_records) - 0.5))
     2                * 360.0 )
        else
          type *, ' ERROR IN EMITTER ROTATION FLAG VALUE '
      end if
c
c Thetad is in degrees CCW from directly toward rcvr; ISAMP is the
c sample number in increments of 0.2 microseconds (5 MHz sample rate).
c There are 3000 points max in a record
c
      do isamp = kstart, num_samples
        if (buffer (isamp, iscan) .ge. thresh)then    !We have a splash
c
c This is the corresponding time delay. A 7.5 % correction has been
```

```
c added to the time delay.
c
              tauv = 0.2 *
     1               float (isamp + nint ((7.5/100.0) * float (isamp)))
c
c
              if ( (tauv - oldtau) .ge. delta_tau ) then
                 ntau = ntau + 1
                 eatau (ntau) = tauv
                 oldtau = tauv
              end if
           end if
        end do
c
        return
        end
```

```
XXXXXXXXXX   333333333333333333333333333333333333333333333333   XXXXXXXXXX
XXXXXXXXXX   33333333  Interactive System Design Center  33333333   XXXXXXXXXX
XXXXXXXXXX   333333333333333333333333333333333333333333333333   XXXXXXXXXX
```

NOTE: JIM

```
  000   RRRR    SSSS  BBBB    000   SSSS
 0   0  R   R  S      B   B  0   0  S
 0   0  R   R  S      B   B  0   0  S
 0   0  RRRR    SSS   BBBB   0   0   SSS
 0   0  R R        S  B   B  0   0      S
 0   0  R  R       S  B   B  0   0      S
  000   R   R  SSSS   BBBB    000   SSSS


  GGGG   000   RRRR   FFFFF
 G      0   0  R   R  F
 G      0   0  R   R  F
 G      0   0  RRRR   FFFF
 G GGG  0   0  R R    F
 G   G  0   0  R  R   F
  GGG    000   R   R  F

FFFFF   000   RRRR     ;;    1    666
F      0   0  R   R    ;;   11   6
F      0   0  R   R          1   6
FFFF   0   0  RRRR     ;;    1   6666
F      0   0  R R      ;;    1   6   6
F      0   0  R  R      ;    1   6   6
F       000   R   R    ;   111    666
```

File _9DRB5:[ORSBOS.PEL]GORF.FOR;16 (790,327,0), last revised on 30-NOV-1987 09:26, is a 11 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 72 bytes.

Job GORF (183) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:28 from queue SPOCK_LAS2.

```
XXXXXXXXXX   333333333333333333333333333333333333333333333333   XXXXXXXXXX
XXXXXXXXXX   Digital Equipment Corporation - VAX/VMS Version V4.6   XXXXXXXXXX
XXXXXXXXXX   333333333333333333333333333333333333333333333333   XXXXXXXXXX
```

```
      SUBROUTINE GORF (MAXK,NCOUNT,MMAXK,INDEX,DISC,MAPCNT,COORDI,
     +    COORDJ,KCHAR,MAXVAL,DISTANCE_FROM_E,IRES,POWER,TITLE,K_E,
     +    MAXPG)
C**********************************************************************C
C                                                                      C
C      Subroutine GORF - Revision 5/2/86                               C
C                                                                      C
C                                                                      C
C      Input Files: None.                                              C
C                                                                      C
C      Output Files:                                                   C
C                                                                      C
C        * Unit 2 - Not opened, received from calling routine.         C
C                                                                      C
C      Routines Used: None.                                            C
C                                                                      C
C**********************************************************************C
      CHARACTER SYMBOL*2,TITLE*106
      INTEGER*4 KK,KFLAG,MAXK,NCOUNT,MMAXK,II,JJ,n1,n2,KCHAR(4),IRES,
     +    K_E,LCOUNT,PCOUNT,MAXPG,MAXPGT,MAXLC1,MAXLC2
      INTEGER*2 INDEX(10000),RANK,MAPCNT(10000),COORDI(10000),
     +    COORDJ(10000),MAXVAL(10000)
      REAL*4 ZK,PERCENTILE,LEVEL,DISC(10000),DISTANCE_FROM_E(10000),
     +    POWER
       INTEGER * 2      ILIM (-1:1), JLIM (2, 2238)
C
      DATA MAXLC1,MAXLC2/52,54/
C**********************************************************************C
C**********************************************************************C
C
      WRITE (2,1000) TITLE
 1000 FORMAT ('1'/' ',A80)
C
C ******  If MAXPG is 0, set to print "all":
C
      IF (MAXPG.EQ.0) THEN
                    MAXPGT=1+NINT(0.5+FLOAT(MMAXK-MAXLC1)/FLOAT(MAXLC2))
                    IF (MAXPGT.LE.0) MAXPGT=1
                      ELSE
                        MAXPGT=MAXPG
                      END IF
C
      WRITE (2,1005)
 1005 FORMAT ('0'/' ',
     +          '                          Average  Maximum',
     +          '   Number    Coordinates       Distance ',
     +    /' ','Index  Rank   %-tile   MD**Power     MD    ',
     +          '  of MD''s        (I,J)            from E  '/' ')
C
      ZK = 100.0/FLOAT(MMAXK)
C
C ******  Go through the sector points in decreasing order of Rank:
C
      KFLAG = 0
      LCOUNT=1
      PCOUNT=1
      DO 200 RANK = MMAXK, 1, -1
      PERCENTILE = ZK*FLOAT(RANK)
C
      KK = INDEX(RANK+NCOUNT)
      IF (MAPCNT(KK).GT.0) THEN
```

```
                                LEVEL = 10.0**(-DISC(KK))-1.0
                              ELSE
                                LEVEL = 0.0
                              END IF
      JJ = COORDJ(KK)
      II = COORDI(KK)
C
      IF (KK.EQ.K_E) THEN
                        SYMBOL='Em'
                        KFLAG=KFLAG+1
      ELSE IF (KK.EQ.KCHAR(1)) THEN
                                  SYMBOL = '--'
                                  KFLAG = KFLAG + 1
      ELSE IF (KK.EQ.KCHAR(2)) THEN
                                  SYMBOL = '-+'
                                  KFLAG = KFLAG + 1
      ELSE IF (KK.EQ.KCHAR(3)) THEN
                                  SYMBOL = '+-'
                                  KFLAG = KFLAG + 1
      ELSE IF (KK.EQ.KCHAR(4)) THEN
                                  SYMBOL = '++'
                                  KFLAG = KFLAG + 1
      ELSE
          SYMBOL='  '
      END IF
C
      n1 = 1 + max (0,ifix (alog10 (0.1 + float (II))))
      n2 = 1 + max (0,ifix (alog10 (0.1 + float (JJ))))
      WRITE (2,1010) KK,RANK,PERCENTILE,LEVEL,MAXVAL(KK),MAPCNT(KK),
     +    II,JJ,SYMBOL,DISTANCE_FROM_E(KK)/1000.0
 1010 FORMAT (' ',I4,3X,I4,3X,F5.1,X,F10.2,2X,I6,3X,I6,<10-n1>X,'(',
     +    I<n1>,',',I<n2>,')',<6-n2>X,A2,3X,F6.2)
C
      IF ((KFLAG.EQ.4).OR.
     +          ((LCOUNT.EQ.MAXLC2).AND.(PCOUNT.EQ.MAXPGT))) GO TO 9999
C
      IF (PCOUNT.EQ.1) THEN
          IF (LCOUNT.EQ.MAXLC1) THEN
                                   WRITE (2,1015)
                                   LCOUNT=1
                                   PCOUNT=PCOUNT+1
                                 ELSE
                                   LCOUNT=LCOUNT+1
                                 END IF
                     ELSE
          IF (LCOUNT.EQ.MAXLC2) THEN
                                   WRITE (2,1015)
                                   LCOUNT=1
                                   PCOUNT=PCOUNT+1
                                 ELSE
                                   LCOUNT=LCOUNT+1
                                 END IF
                     END IF
 1015 FORMAT ('1'/'0',
     +         '                        Average  Maximum',
     +         '  Number    Coordinates       Distance ',
     +    /' ','Index  Rank   %-tile   MD**Power     MD    ',
     +         '  of MD''s       (I,J)           from E   '/' ')
C
  200 CONTINUE
C
```

```
C
 9999 RETURN
      END
```

```
YYYYYYYYYY    444444444444    44444444444444444444444444444444    YYYYYYYYYY
YYYYYYYYYY    44444444  Interactive System Design Center  44444444    YYYYYYYYYY
YYYYYYYYYY    4444444444444444444444444444444444444444444444444444    YYYYYYYYYY

NOTE: JIM

                     000   RRRR    SSSS  BBBB    000    SSSS
                    0   0  R   R  S      B   B  0   0  S
                    0   0  R   R  S      B   B  0   0  S
                    0   0  RRRR    SSS   BBBB   0   0   SSS
                    0   0  R R        S  B   B  0   0      S
                    0   0  R  R       S  B   B  0   0      S
                     000   R   R  SSSS   BBBB    000    SSSS

                H   H  III   SSSS  TTTTT   000    CCCC    1
                H   H   I   S        T    0   0  C       11
                H   H   I   S        T    0   0  C        1
                HHHHH   I    SSS     T    0   0  C        1
                H   H   I       S    T    0   0  C        1
                H   H   I       S    T    0   0  C        1
                H   H  III  SSSS     T     000    CCCC   111

                       FFFFF   000   RRRR      ;;    1    7777
                       F      0   0  R   R     ;;   11       7
                       F      0   0  R   R           1       7
                       FFFF   0   0  RRRR      ;;    1      7
                       F      0   0  R R       ;;    1     7
                   ..  F      0   0  R  R       ;    1    7
                   ..  F       000   R   R     ;    111  7
```

File _9DRB5:[ORSBOS.PEL]HISTOC1.FOR;17 (791,348,0), last revised on 30-NOV-1987 09:26, is a 16 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 72 bytes.

Job HISTOC1 (184) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:28 from queue SPOCK_LAS2.

```
YYYYYYYYYY    4444444444444444444444444444444444444444444444444444    YYYYYYYYYY
YYYYYYYYYY    Digital Equipment Corporation - VAX/VMS Version V4.6    YYYYYYYYYY
YYYYYYYYYY    4444444444444444444444444444444444444444444444444444    YYYYYYYYYY
```

```
      SUBROUTINE HISTOC1 (I[illegible],MINDISC,MAXDISC,DISC,IRES,MAP[illegible],SCORE,
     +    K_E,IDN,IUP,ILIM,JLIM)
C*********************************************************************C
C                                                                     C
C     Subroutine HISTOC1 - Revision 3/12/86                           C
C                                                                     C
C                                                                     C
C     Input Files: None.                                              C
C                                                                     C
C     Output Files: None.                                             C
C                                                                     C
C       * Unit 2 - Not opened, from caller.                           C
C                                                                     C
C     Routines Used: None.                                            C
C                                                                     C
C*********************************************************************C
      CHARACTER OLINE*106,HLINE(106)*1,SYMBOL*1
      INTEGER*2 MAPCNT(10000),ILIM(-1:1),JLIM(2,IDN:IUP)
      INTEGER*4 ITER,IRES,KK,IO_R,JO_R,LL,K_E,ILEN,IDN,IUP
      REAL*4 LEVEL,SCALE,MINDISC,MAXDISC,DISC(10000),SCORE
C
C
      DATA HLINE/106*'+'/
C
C ******  COMMON blocks:
C
      CHARACTER BLANKL*106,TITLE*106
      INTEGER*4 JMINP,JMAXP,IMINP,IMAXP,JLEN
      COMMON /HIST/ BLANKL,TITLE,JMINP,JMAXP,IMINP,IMAXP,JLEN
C
      INTEGER*4 IO_A,JO_A,IO_A_M,JO_A_M,IO_B,JO_B,IO_B_M,JO_B_M,
     +    IO_E,JO_E,IO_E_M,JO_E_M
      COMMON /POSITS/ IO_A,JO_A,IO_A_M,JO_A_M,IO_B,JO_B,IO_B_M,JO_B_M,
     +    IO_E,JO_E,IO_E_M,JO_E_M
C*********************************************************************C
C*********************************************************************C
C
C ******  Print page headers:
C
      WRITE (2,1005) TITLE
 1005 FORMAT ('1'/' ',A80)
C
      WRITE (2,1010) ITER
 1010 FORMAT ('0'/' ','Cumulative location estimate after ',I2,
     +    ' scans:')
C
C ******  Print limits of the "value" and that at the closest point
C ******  to the emitter:
C
      WRITE (2,1015) MINDISC, MAXDISC
 1015 FORMAT ('0'/' ',4X,'Minimum discriminant =',F7.3,';  maximum ',
     +     'discriminant = ',F7.3)
C
      IF (K_E.GE.1) THEN
                  WRITE (2,1013) DISC(K_E)
                ELSE
                  WRITE (2,1045)
                END IF
 1013 FORMAT ('0',4X,'Discriminant of map point nearest Emitter: ',F7.3)
 1045 FORMAT ('0',4X,'Map point nearest Emitter is not in the sector')
C
      SCALE = 100.0/(MAXDISC-MINDISC)
```

```
C
      IF (K_E.GE.1) THEN
                 IF (MAPCNT(K_E).LE.0) THEN
                                            LEVEL=0.0
                                          ELSE
                                            LEVEL=SCALE*(DISC(K_E)-MINDISC)
                                          END IF
C
      IF (LEVEL.GE.80.0) THEN
                              SYMBOL = '#'
      ELSE IF (LEVEL.GE.60.0) THEN
                                   SYMBOL = '*'
      ELSE IF (LEVEL.GE.40.0) THEN
                                   SYMBOL = '+'
      ELSE IF (LEVEL.GE.20.0) THEN
                                   SYMBOL = '.'
      ELSE
          SYMBOL = ' '
      END IF
C
      WRITE (2,1020) LEVEL,SYMBOL
 1020 FORMAT ('0',4X,'Level of map point nearest E (100 = max, ',
     +    '0 = min):  ',F6.2,'  (',A1,')')
                  END IF
C
C ******  Print map score:
C
      WRITE (2,1000) SCORE
 1000 FORMAT ('0'/' ',4X,'Overall Map Score = ',F9.3)
C
C ******  Print histogram legend:
C
      WRITE (2,1030)
 1030 FORMAT ('0',
     +    /'0','Legend:  #    denotes  80% <= Level <= 100%',
     +    /' ','          *    denotes  60% <= Level <   80%',
     +    /' ','          +    denotes  40% <= Level <   60%',
     +    /' ','          .    denotes  20% <= Level <   40%',
     +    /' ','        blank  denotes   0% <= Level <   20%')
C
C ******  Print Histogram; first, the header(s):
C
      WRITE (2,1035) JMINP,JMAXP,IRES
 1035 FORMAT ('0'/' ',8X,'(J values are from ',I4,' to ',I4,' by ',I4,
     +    ')'/'0')
C
C ******  If too close to bottom of page, go to next page:
C
      ILEN=1+NINT(FLOAT(IMAXP-IMINP)/FLOAT(IRES))
      IF (ILEN.GT.29) WRITE (2,1055)
 1055 FORMAT ('1')
C
C ******  Start of actual map:
C
      WRITE (2,1025) (HLINE(JJ),JJ=1,JLEN)
 1025 FORMAT (' ',6X,106A1)
C
C ******  Part of map "above" processing sector:
C
      DO 210 IO_R = IMINP,ILIM(-1)-1,IRES
      OLINE = BLANKL
```

```
C
C      IF (IO_R.EQ.IO_B_M) THEN
C                               LL=1+(JO_B_M-JMINP)/IRES
C                               OLINE(LL:LL)='B'
C                            END IF
C
C      IF (IO_R.EQ.IO_A_M) THEN
C                                 LL=1+(JO_A_M-JMINP)/IRES
C                                 OLINE(LL:LL)='A'
C                               END IF
C
      IF (IO_R.EQ.IO_E_M) THEN
                                LL=1+(JO_E_M-JMINP)/IRES
                                OLINE(LL:LL)='E'
                              END IF
C
      WRITE (2,1040) IO_R, OLINE(1:JLEN)
  210 CONTINUE
C
C ******  Part of map including processing sector:
C
      KK = 0
      DO 200 IO_R = ILIM(-1), ILIM(1), IRES
      OLINE = BLANKL
C
      DO 300 JO_R = JLIM(1,IO_R), JLIM(2,IO_R), IRES
      KK = KK + 1
C
      IF (MAPCNT(KK).LE.0) THEN
                                 LEVEL=0.0
                               ELSE
                                 LEVEL=SCALE*(DISC(KK)-MINDISC)
                               END IF
C
      IF (LEVEL.GE.80.0) THEN
                               SYMBOL = '#'
      ELSE IF (LEVEL.GE.60.0) THEN
                                     SYMBOL = '*'
      ELSE IF (LEVEL.GE.40.0) THEN
                                     SYMBOL = '+'
      ELSE IF (LEVEL.GE.20.0) THEN
                                     SYMBOL = '.'
      ELSE
          SYMBOL = ' '
      END IF
C
      LL = 1+(JO_R-JMINP)/IRES
      OLINE(LL:LL) = SYMBOL
  300 CONTINUE
C
C      IF (IO_R.EQ.IO_B_M) THEN
C                               LL=1+(JO_B_M-JMINP)/IRES
C                               OLINE(LL:LL)='B'
C                            END IF
C
C      IF (IO_R.EQ.IO_A_M) THEN
C                                 LL=1+(JO_A_M-JMINP)/IRES
C                                 OLINE(LL:LL)='A'
C                               END IF
C
      IF (IO_R.EQ.IO_E_M) THEN
```

```
                                        1+(JO_E_M-JMINP)/IRES
                                    OLINE(LL:LL)='E'
                                END IF
C
      WRITE (2,1040) IO_R, OLINE(1:JLEN)
 1040 FORMAT (' ',I3,2X,'+',A<JLEN>,'+')
  200 CONTINUE
C
C ******  Part of map "below" processing sector:
C
      DO 220 IO_R = ILIM(1)+1,IMAXP,IRES
      OLINE = BLANKL
C
C      IF (IO_R.EQ.IO_B_M) THEN
C                                LL=1+(JO_B_M-JMINP)/IRES
C                                OLINE(LL:LL)='B'
C                             END IF
C
C      IF (IO_R.EQ.IO_A_M) THEN
C                                  LL=1+(JO_A_M-JMINP)/IRES
C                                  OLINE(LL:LL)='A'
C                               END IF
C
      IF (IO_R.EQ.IO_E_M) THEN
                                 LL=1+(JO_E_M-JMINP)/IRES
                                 OLINE(LL:LL)='E'
                              END IF
C
      WRITE (2,1040) IO_R, OLINE(1:JLEN)
  220 CONTINUE
C
C ******  "Bottom header:"
C
      WRITE (2,1025) (HLINE(JJ),JJ=1,JLEN)
C
C ******  Return to caller:
C
      RETURN
      END
```

```
ZZZZZZZZZZ    5555555555555 5555555555555555555555555555555  5555    ZZZZZZZZZZ
ZZZZZZZZZZ    55555555  Interactive System Design Center  55555555    ZZZZZZZZZZ
ZZZZZZZZZZ    5555555555555555555555555555555555555555555555555555    ZZZZZZZZZZ
```

NOTE: JIM

```
         000    RRRR     SSSS   BBBB    000    SSSS
        0   0   R   R   S       B   B  0   0  S
        0   0   R   R   S       B   B  0   0  S
        0   0   RRRR     SSS    BBBB   0   0   SSS
        0   0   R R         S   B   B  0   0      S
        0   0   R  R        S   B   B  0   0      S
         000    R   R   SSSS    BBBB    000   SSSS

   H   H   III    SSSS  TTTTT   000    CCCC   222
   H   H    I    S        T    0   0  C      2   2
   H   H    I    S        T    0   0  C          2
   HHHHH    I     SSS     T    0   0  C         2
   H   H    I        S    T    0   0  C        2
   H   H    I        S    T    0   0  C       2
   H   H   III   SSSS     T     000    CCCC  22222

        FFFFF   000    RRRR     ;;    222    000
        F      0   0   R   R    ;;   2   2  0   0
        F      0   0   R   R             2  0  00
        FFFF   0   0   RRRR     ;;      2   0 0 0
        F      0   0   R R      ;;     2    00  0
    ..  F      0   0   R  R      ;    2     0   0
    ..  F       000    R   R    ;    22222   000
```

File 9DRB5:[ORSBOS.PEL]HISTOC2.FOR;20 (792,349,0), last revised on 30-NOV-1987 09:26, is a 18 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 72 bytes.

Job HISTOC2 (185) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:29 from queue SPOCK_LAS2.

```
ZZZZZZZZZZ    5555555555555555555555555555555555555555555555555555    ZZZZZZZZZZ
ZZZZZZZZZZ    Digital Equipment Corporation - VAX/VMS Version V4.6    ZZZZZZZZZZ
ZZZZZZZZZZ    5555555555555555555555555555555555555555555555555555    ZZZZZZZZZZ
```

```
      SUBROUTINE HISTOC2 (I'   ,RANK,MMAXK,MINDISC,MAXDISC,DI   IRES,
     +    MAPCNT,RFLAG,SCORE,K_E,IDN,IUP,ILIM,JLIM)
C*********************************************************************C
C                                                                     C
C     Subroutine HISTOC2 - Revision 3/12/86                           C
C                                                                     C
C                                                                     C
C     Input Files: None.                                              C
C                                                                     C
C     Output Files: None.                                             C
C                                                                     C
C       * Unit 2 - Not opened, from caller.                           C
C                                                                     C
C     Routines Used: None.                                            C
C                                                                     C
C*********************************************************************C
      LOGICAL RFLAG
      CHARACTER OLINE*106,HLINE(106)*1,SYMBOL*1
      INTEGER*2 MAPCNT(10000),RANK(10000),ILIM(-1:1),
     +    JLIM(2,IDN:IUP)
      INTEGER*4 ITER,MMAXK,IO_R,JO_R,KK,LL,IRES,JJ,K_E,ILEN,IDN,IUP
      REAL*4 LEVEL,SCALE,MINDISC,MAXDISC,DISC(10000),SCORE
C
      DATA HLINE/106*'+'/
C
C ******  COMMON blocks:
C
      CHARACTER BLANKL*106,TITLE*106
      INTEGER*4 JMINP,JMAXP,IMINP,IMAXP,JLEN
      COMMON /HIST/ BLANKL,TITLE,JMINP,JMAXP,IMINP,IMAXP,JLEN
C
      INTEGER*4 IO_A,JO_A,IO_A_M,JO_A_M,IO_B,JO_B,IO_B_M,JO_B_M,
     +    IO_E,JO_E,IO_E_M,JO_E_M
      COMMON /POSITS/ IO_A,JO_A,IO_A_M,JO_A_M,IO_B,JO_B,IO_B_M,JO_B_M,
     +    IO_E,JO_E,IO_E_M,JO_E_M
C*********************************************************************C
C*********************************************************************C
C
C ******  Print page headers:
C
      WRITE (2,1005) TITLE
 1005 FORMAT ('1'/' ',A80)
C
      WRITE (2,1010) ITER
 1010 FORMAT ('0'/' ','Cumulative location estimate after ',I4,
     +    ' scans:')
C
C ******  Print limits of the "value" and that at the closest point
C ******  to the emitter:
C
      IF (RFLAG) THEN
                 WRITE (2,1014) MMAXK
               ELSE
                 WRITE (2,1015) MINDISC, MAXDISC
               END IF
 1014 FORMAT ('0'/' ',4X,'Highest ranking is: ',I5)
C
 1015 FORMAT ('0'/' ',4X,'Minimum discriminant =',F7.3,';  maximum ',
     +    'discriminant = ',F7.3)
C
      IF (K_E.GE.1) THEN
```

```
                    IF (F  AG) THEN
                              WRITE (2,1012) RANK(K_E)
                            ELSE
                              WRITE (2,1013) DISC(K_E)
                            END IF
                  ELSE
                    WRITE (2,1045)
                  END IF
 1012 FORMAT ('0',4X,'Ranking of map point nearest Emitter is: ',I5)
 1013 FORMAT ('0',4X,'Discriminant of map point nearest Emitter: ',F7.3)
 1045 FORMAT ('0',4X,'Map point nearest Emitter is not in the sector')
C
C ******  Calculate and print "Level" at point closest to emitter:
C
      IF (RFLAG) THEN
                  SCALE = 100.0/FLOAT(MMAXK)
                ELSE
                  SCALE = 100.0/(MAXDISC-MINDISC)
                END IF
C
      IF (K_E.GE.1) THEN
      IF (RFLAG) THEN
                  LEVEL = SCALE*RANK(K_E)
                ELSE
                  IF (MAPCNT(K_E).LE.0) THEN
                                    LEVEL = 0.0
                                    ELSE
                                    LEVEL=SCALE*(DISC(K_E)-MINDISC)
                                    END IF
                END IF
C
      IF (LEVEL.GE.99.0) THEN
                        SYMBOL = '#'
      ELSE IF (LEVEL.GE.95.0) THEN
                            SYMBOL = '*'
      ELSE IF (LEVEL.GE.85.0) THEN
                            SYMBOL = '+'
      ELSE IF (LEVEL.GE.60.0) THEN
                            SYMBOL = '.'
      ELSE
         SYMBOL = ' '
      END IF
C
      WRITE (2,1020) LEVEL,SYMBOL
 1020 FORMAT ('0',4X,'Level of map point nearest E (100 = max, ',
     +    '0 = min):  ',F6.2,'  (',A1,')')
                    END IF
C
C ******  Print map score:
C
      WRITE (2,1000) SCORE
 1000 FORMAT ('0'/' ',4X,'Overall Map Score = ',F9.3)
C
C ******  Print histogram legend:
C
      WRITE (2,1030)
 1030 FORMAT ('0',
     +    /'0','Legend:  #    denotes  99% <= Level <= 100%',
     +    /' ','          *    denotes  95% <= Level <   99%',
     +    /' ','          +    denotes  85% <= Level <   95%',
     +    /' ','          .    denotes  60% <= Level <   85%',
```

```
     +    /' ',' blar  denotes  0% <= Level <  60%'
C
C ****** Print Histogram; first, the header(s):
C
      WRITE (2,1035) JMINP,JMAXP,IRES
 1035 FORMAT ('0'/' ',8X,'(J values are from ',I4,' to ',I4,' by ',I4,
     +    ')'/'0')
C
C ****** If too close to bottom of page, go to next page:
C
      ILEN=1+NINT(FLOAT(IMAXP-IMINP)/FLOAT(IRES))
      IF (ILEN.GT.29) WRITE (2,1055)
 1055 FORMAT ('1')
C
C ****** Start of actual "map":
C
      WRITE (2,1025) (HLINE(JJ),JJ=1,JLEN)
 1025 FORMAT (' ',6X,106A1)
C
C ****** Part of map "above" processing sector:
C
      DO 210 IO_R = IMINP,ILIM(-1)-1,IRES
      OLINE = BLANKL
C
C      IF (IO_R.EQ.IO_B_M) THEN
C                         LL=1+(JO_B_M-JMINP)/IRES
C                         OLINE(LL:LL)='B'
C                       END IF
C
C      IF (IO_R.EQ.IO_A_M) THEN
C                           LL=1+(JO_A_M-JMINP)/IRES
C                           OLINE(LL:LL)='A'
C                         END IF
C
      IF (IO_R.EQ.IO_E_M) THEN
                          LL=1+(JO_E_M-JMINP)/IRES
                          OLINE(LL:LL)='E'
                        END IF
C
      WRITE (2,1040) IO_R, OLINE(1:JLEN)
  210 CONTINUE
C
C ****** Part of map including processing sector:
C
      KK = 0
      DO 200 IO_R = ILIM(-1), ILIM(1), IRES
      OLINE = BLANKL
C
      DO 300 JO_R = JLIM(1,IO_R), JLIM(2,IO_R), IRES
      KK = KK + 1
C
      IF (RFLAG) THEN
                   LEVEL = SCALE * RANK(KK)
                 ELSE
                   IF (MAPCNT(KK).LE.0) THEN
                                          LEVEL=0.0
                                        ELSE
                                          LEVEL=SCALE*(DISC(KK)-MINDISC)
                                        END IF
                 END IF
C
```

```
      IF (LEVEL.GE.99.0) TI
                                    SYMBOL = '#'
      ELSE IF (LEVEL.GE.95.0) THEN
                                    SYMBOL = '*'
      ELSE IF (LEVEL.GE.85.0) THEN
                                    SYMBOL = '+'
      ELSE IF (LEVEL.GE.60.0) THEN
                                    SYMBOL = '.'
      ELSE
          SYMBOL = ' '
      END IF
C
      LL=1+(JO_R-JMINP)/IRES
      OLINE(LL:LL) = SYMBOL
c
c WRITE OUT LEVEL VALUES TO FILE FOR 3D PLOTS
c
c       write (80, *) level
c
 300 CONTINUE
C
C      IF (IO_R.EQ.IO_B_M) THEN
C                                LL=1+(JO_B_M-JMINP)/IRES
C                                OLINE(LL:LL)='B'
C                              END IF
C
C      IF (IO_R.EQ.IO_A_M) THEN
C                                 LL=1+(JO_A_M-JMINP)/IRES
C                                 OLINE(LL:LL)='A'
C                               END IF
C
      IF (IO_R.EQ.IO_E_M) THEN
                                LL=1+(JO_E_M-JMINP)/IRES
                                OLINE(LL:LL)='E'
                              END IF
C
      WRITE (2,1040) IO_R, OLINE(1:JLEN)
 1040 FORMAT (' ',I3,2X,'+',A<JLEN>,'+')
  200 CONTINUE
C
C****** Part of map "below" processing sector:
C
      DO 220 IO_R = ILIM(1)+1,IMAXP,IRES
      OLINE = BLANKL
C
C      IF (IO_R.EQ.IO_B_M) THEN
C                                LL=1+(JO_B_M-JMINP)/IRES
C                                OLINE(LL:LL)='B'
C                              END IF
C
C      IF (IO_R.EQ.IO_A_M) THEN
C                                 LL=1+(JO_A_M-JMINP)/IRES
C                                 OLINE(LL:LL)='A'
C                               END IF
C
      IF (IO_R.EQ.IO_E_M) THEN
                                LL=1+(JO_E_M-JMINP)/IRES
                                OLINE(LL:LL)='E'
                              END IF
C
      WRITE (2,1040) IO_R, OLINE(1:JLEN)
```

```
 220 CONTINUE
C
C ******  "Bottom header:"
C
      WRITE (2,1025) (HLINE(JJ),JJ=1,JLEN)
C
C ******  Return to caller:
C
c
c
c
c        write (80, 500)
c500      FORMAT (' /')
      RETURN
      END
```

```
AAAAAAAAAA     6666666666666 66666666666666666666666666666666 6666     AAAAAAAAAA
AAAAAAAAAA     66666666  Interactive System Design Center  66666666    AAAAAAAAAA
AAAAAAAAAA     666666666666666666666666666666666666666666666666666     AAAAAAAAAA
```

NOTE: JIM

```
   000   RRRR    SSSS  BBBB    000    SSSS
  0   0  R   R  S      B   B  0   0  S
  0   0  R   R  S      B   B  0   0  S
  0   0  RRRR    SSS   BBBB   0   0   SSS
  0   0  R R        S  B   B  0   0      S
  0   0  R  R       S  B   B  0   0      S
   000   R   R  SSSS   BBBB    000   SSSS

L      III  N   N  H   H  III   SSSS  TTTTT
L       I   N   N  H   H   I   S        T
L       I   NN  N  H   H   I   S        T
L       I   N N N  HHHHH   I    SSS     T
L       I   N  NN  H   H   I       S    T
L       I   N   N  H   H   I       S    T
LLLLL  III  N   N  H   H  III  SSSS     T

FFFFF   000   RRRR    ;;    1    4   4
F      0   0  R   R   ;;   11    4   4
F      0   0  R   R         1    4   4
FFFF   0   0  RRRR    ;;    1    44444
F      0   0  R R     ;;    1        4
F      0   0  R  R     ;    1        4
F       000   R   R   ;   111        4
```

File _9DRB5:[ORSBOS.PEL]LINHIST.FOR;14 (793,288,0), last revised on 30-NOV-1987 09:26, is a 14 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 72 bytes.

Job LINHIST (186) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:29 from queue SPOCK_LAS2.

```
AAAAAAAAAA     666666666666666666666666666666666666666666666666666     AAAAAAAAAA
AAAAAAAAAA     Digital Equipment Corporation - VAX/VMS Version V4.6    AAAAAAAAAA
AAAAAAAAAA     666666666666666666666666666666666666666666666666666     AAAAAAAAAA
```

```
      SUBROUTINE LINHIST (I   R,TITLE,MAXK,K_E,MINDISC,MAXDI    DISC,
     +    MAPCNT)
C************************************************************************C
C                                                                        C
C     Subroutine LINHIST - Revision 3/14/86-A                            C
C                                                                        C
C         This subroutine prints, to Unit 2, a bar graph type histo-     C
C     gram of the discriminant values.                                   C
C         The discriminant values are stored in the vector DISC, and     C
C     there are MAXK values. However, if a given element of the vector   C
C     MAPCNT (map counters) is less than or equal to zero, the corre-    C
C     sponding value of DISC is ignored. (Elsewhere there is a           C
C     variable called NCOUNT which is the number of these invalid        C
C     values and a variable called MMAXK which is the difference, the    C
C     number of valid values.)                                           C
C         ITER is just the number of "scans" that have been processed    C
C     so far, and is used in the header line.                            C
C         The parameters MAXDISC and MINDISC are the maximum and         C
C     minimum values from among the (valid) elements of DISC.            C
C         This bar graph starts on a new page, and takes only one page   C
C     (58 lines) and is 80 characters wide.                              C
C         No external variable values are altered by this routine.      C
C                                                                        C
C------------------------------------------------------------------------C
C                                                                        C
C     Input Files: None.                                                 C
C                                                                        C
C     Output Files: None.                                                C
C                                                                        C
C       * Unit 2 - Not opened, from caller.                              C
C                                                                        C
C     Routines Used: None.                                               C
C                                                                        C
C************************************************************************C
      CHARACTER BAR(50)*1,POINTER(50)*1,TITLE*106
      INTEGER*2 MAPCNT(10000)
      INTEGER*4 ITER,MAXK,SEGCNT(50),ISEG,KK,BLEN,II,MAXCNT,K_E
      REAL*4 MINDISC,MAXDISC,DISC(10000),SEGD(49),SCALE
C
      DATA BAR/50*'*'/

C************************************************************************C
C
C ******  Initialize segment boundaries and counters:
C
      DO 100 ISEG=1,49
      SEGD(ISEG)=MINDISC+(MAXDISC-MINDISC)*(FLOAT(ISEG)/50.0)
      SEGCNT(ISEG)=0
  100 CONTINUE
      SEGCNT(50)=0
C
C ******  Locate Emitter value; if K_E is -1, Emitter is not in the
C ******  sector and thus has no discriminant. If MAPCNT(K_E) is -1,
C ******  then Emitter is within RMIN of the aircraft, and has no
C ******  discriminant. And if MAPCNT(K_E) is 0, discriminant is too
C ******  large (negative) to use.
C
      DO 150 ISEG=1,50
      POINTER(ISEG)=' '
  150 CONTINUE
C
```

```
      IF (K_E.GE.1) THEN
         IF (MAPCNT(K_E).GT.0) THEN
                          DO ISEG=1,49
                          IF (DISC(K_E).LT.SEGD(ISEG)) THEN
                                      POINTER(ISEG)='>'
                                      GO TO 170
                                          END IF
                          END DO
                          POINTER(50)='>'
                        END IF
              END IF
C
C ******  Compute the histogram:
C
  170 DO 200 KK=1,MAXK
      IF (MAPCNT(KK).LE.0) GO TO 200
      DO 250 ISEG=1,49
      IF (DISC(KK).LT.SEGD(ISEG)) THEN
                           SEGCNT(ISEG)=SEGCNT(ISEG)+1
                           GO TO 200
                          END IF
  250 CONTINUE
      SEGCNT(50)=SEGCNT(50)+1
  200 CONTINUE
C
C ******  Histogram #1:
C
C ******  Print Header:
C
C      WRITE (2,1000) TITLE
 1000 FORMAT('1'/' ',A80)
C
C      WRITE (2,1005) ITER
 1005 FORMAT ('0'/' ','Culmulative Discriminant histogram after ',I3,
     +    ' scans:')
C
C ******  Find the maximum count from among the segments:
C
      MAXCNT=SEGCNT(1)
      DO 350 ISEG=2,50
      MAXCNT=MAX(MAXCNT,SEGCNT(ISEG))
  350 CONTINUE
C
C ******  If no points had any hits, don't produce a bar graph:
C
      IF (MAXCNT.EQ.0) THEN
                     WRITE (2,1020)
                     GO TO 9999
                    END IF
 1020 FORMAT ('0'/'0','****  All points had no hits.')
C
C ******  Print the histogram:
C
C      SCALE=50.0/FLOAT(MAXCNT)
C
C      WRITE (2,1015)
 1015 FORMAT ('0'/' ')
C
C      ISEG=1
C      BLEN=NINT(SCALE*FLOAT(SEGCNT(ISEG)))
C      WRITE (2,1010) ISEG,MINDISC,SEGD(ISEG),SEGCNT(ISEG),
```

```
C     +     POINTER(ISEG),(B   II),II=1,BLEN)
C
C       DO 300 ISEG=2,49
C       BLEN=NINT(SCALE*FLOAT(SEGCNT(ISEG)))
C       WRITE (2,1010) ISEG,SEGD(ISEG-1),SEGD(ISEG),SEGCNT(ISEG),
C     +     POINTER(ISEG),(BAR(II),II=1,BLEN)
C   300 CONTINUE
C
C       ISEG=50
C       BLEN=NINT(SCALE*FLOAT(SEGCNT(ISEG)))
C       WRITE (2,1010) ISEG,SEGD(ISEG-1),MAXDISC,SEGCNT(ISEG),
C     +     POINTER(ISEG),(BAR(II),II=1,BLEN)
C
 1010 FORMAT (' ',I2,': ',F7.3,' - ',F7.3,' (',I4,')',A1,'|',50A1)
C
C ******  Histogram #2:
C
C ******  Print Header:
C
      WRITE (2,1000) TITLE
C
      WRITE (2,1005) ITER
C
C ******  Print the histogram:
C
      SCALE=1.0
C
      WRITE (2,1015)
C
      ISEG=1
      BLEN=NINT(SCALE*FLOAT(SEGCNT(ISEG)))
      IF (BLEN.GT.50) BLEN=50
      WRITE (2,1010) ISEG,MINDISC,SEGD(ISEG),SEGCNT(ISEG),
     +     POINTER(ISEG),(BAR(II),II=1,BLEN)
C
      DO 400 ISEG=2,49
      BLEN=NINT(SCALE*FLOAT(SEGCNT(ISEG)))
      IF (BLEN.GT.50) BLEN=50
      WRITE (2,1010) ISEG,SEGD(ISEG-1),SEGD(ISEG),SEGCNT(ISEG),
     +     POINTER(ISEG),(BAR(II),II=1,BLEN)
  400 CONTINUE
C
      ISEG=50
      BLEN=NINT(SCALE*FLOAT(SEGCNT(ISEG)))
      IF (BLEN.GT.50) BLEN=50
      WRITE (2,1010) ISEG,SEGD(ISEG-1),MAXDISC,SEGCNT(ISEG),
     +     POINTER(ISEG),(BAR(II),II=1,BLEN)
C
C ******  Return to caller:
C
 9999 RETURN
      END
```

```
CCCCCCCCCC   88888888888888  88888888888888888888888888888  88888   CCCCCCCCCC
CCCCCCCCCC   88888888  Interactive System Design Center  88888888   CCCCCCCCCC
CCCCCCCCCC   8888888888888888888888888888888888888888888888888888   CCCCCCCCCC
```

NOTE: JIM

```
 000   RRRR    SSSS  BBBB    000    SSSS
0   0  R   R  S      B   B  0   0  S
0   0  R   R  S      B   B  0   0  S
0   0  RRRR    SSS   BBBB   0   0   SSS
0   0  R R        S  B   B  0   0      S
0   0  R  R       S  B   B  0   0      S
 000   R   R  SSSS   BBBB    000   SSSS

 SSSS   CCCC   000   RRRR   EEEEE  FFFFF
S      C      0   0  R   R  E      F
S      C      0   0  R   R  E      F
 SSS   C      0   0  RRRR   EEEE   FFFF
    S  C      0   0  R R    E      F
    S  C      0   0  R  R   E      F
SSSS    CCCC   000   R   R  EEEEE  F

   FFFFF   000   RRRR      ;;    222    000
   F      0   0  R   R     ;;   2   2  0   0
   F      0   0  R   R             2   0  00
   FFFF   0   0  RRRR      ;;     2    0 0 0
   F      0   0  R R       ;;    2     00  0
.. F      0   0  R  R       ;   2      0   0
.. F       000   R   R     ;   22222    000
```

File 9DRB5:[ORSBOS.PEL]SCOREF.FOR;20 (977,433,0), last revised on 30-NOV-1987 09:26, is a 5 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 72 bytes.

Job SCOREF (188) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:36 from queue SPOCK_LAS2.

```
CCCCCCCCCC   8888888888888888888888888888888888888888888888888888   CCCCCCCCCC
CCCCCCCCCC   Digital Equipment Corporation - VAX/VMS Version V4.6   CCCCCCCCCC
CCCCCCCCCC   8888888888888888888888888888888888888888888888888888   CCCCCCCCCC
```

```
      SUBROUTINE SCOREF (DI    ,MINDISC,MAXDISC,MAPCNT,MAXK,MM   K,K_E,
     +    DISTANCE_FROM_E,RCRIT,ALPHAR,ALPHAL,SCORE)
C************************************************************************C
C                                                                        C
C     Subroutine SCOREF - Revision 2/28/86                               C
C                                                                        C
C                                                                        C
C     Input Files: None.                                                 C
C                                                                        C
C     Output Files: None.                                                C
C                                                                        C
C     Routines Used: None.                                               C
C                                                                        C
C************************************************************************C
      INTEGER*2 MAPCNT(10000)
      INTEGER*4 MMAXK,MAXK,K_E,KK
      REAL*4 DISC(10000),MINDISC,MAXDISC,SCORE,ALPHAR,RCRIT,TEMPR,
     +    DISTANCE_FROM_E(10000),TEMPL,ALPHAL
C************************************************************************C
C
C ******  K_E equal to -1 indicates that the emitter is not in the
C ******  sector; set SCORE to a very low value, and exit:
C
      IF (K_E.LE.0) THEN
                        SCORE=-1000.0
                        GO TO 9999
                      END IF
C
C ******  If there were no hits at the Emitter position, then set score
C ******  to a very low value, and exit:
C
      IF (MAPCNT(K_E).LE.0) THEN
                                SCORE=-1000.0
                                GO TO 9999
                              END IF
C
C ******  Sum component scores from each sector point other than that
C ******  nearest to the emitter:
C
      SCORE=0.0
      DO 100 KK=1,MAXK
      IF (MAPCNT(KK).LE.0.OR.KK.EQ.K_E) GO TO 100
C
      TEMPR=(DISTANCE_FROM_E(KK)/RCRIT)
      IF (TEMPR.LT.1.0) TEMPR=1.0
      TEMPR=TEMPR**ALPHAR
C
      TEMPL=(DISC(K_E)-DISC(KK))/ALPHAL
      IF (TEMPL.GE.0.0) THEN
                          SCORE=SCORE+TEMPL/TEMPR
                        ELSE
                          SCORE=SCORE+TEMPL*TEMPR
                        END IF
  100 CONTINUE
C
      SCORE=SCORE*(100.0/FLOAT(MMAXK-1))
C
C ******  Return to caller:
C
 9999 RETURN
      END
```

```
DDDDDDDDDD    9999999999999  999999999999999999999999999999  9999    DDDDDDDDDD
DDDDDDDDDD    99999999  Interactive System Design Center  99999999    DDDDDDDDDD
DDDDDDDDDD    9999999999999999999999999999999999999999999999999999    DDDDDDDDDD

NOTE: JIM

                    000    RRRR     SSSS  BBBB     000     SSSS
                   0   0   R   R   S      B   B   0   0   S
                   0   0   R   R   S      B   B   0   0   S
                   0   0   RRRR     SSS   BBBB    0   0    SSS
                   0   0   R R         S  B   B   0   0       S
                   0   0   R  R        S  B   B   0   0       S
                    000    R   R   SSSS   BBBB     000    SSSS

                           SSSS   EEEEE    CCCC   TTTTT
                          S       E       C         T
                          S       E       C         T
                           SSS    EEEE    C         T
                              S   E       C         T
                              S   E       C         T
                          SSSS    EEEEE    CCCC     T

                      FFFFF   000    RRRR      ;;      1     888
                      F      0   0   R   R     ;;     11    8   8
                      F      0   0   R   R             1    8   8
                      FFFF   0   0   RRRR      ;;      1     888
                      F      0   0   R R       ;;      1    8   8
                   .. F      0   0   R  R       ;      1    8   8
                   .. F       000    R   R     ;      111    888
```

File _9DRB5:[ORSBOS.PEL]SECT.FOR;18 (992,325,0), last revised on 30-NOV-1987 09:26, is a 28 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 74 bytes.

Job SECT (189) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:36 from queue SPOCK_LAS2.

```
DDDDDDDDDD    9999999999999999999999999999999999999999999999999999    DDDDDDDDDD
DDDDDDDDDD    Digital Equipment Corporation - VAX/VMS Version V4.6    DDDDDDDDDD
DDDDDDDDDD    9999999999999999999999999999999999999999999999999999    DDDDDDDDDD
```

```
      SUBROUTINE SECT (RMIN  ) C,JO_C,DISTANCE_FROM_E,KCHAR,   VAL,
     +    COORDI,COORDJ,CUMSUM,IRES,MAPCNT,MAXK,DISTANCE_FROM_B,MMAXK,
     -    NCOUNT,K_E,MAXSEC,ILIM,JLIM,IDN,IUP,JDN,JUP,SECFLG,ERR)
C**********************************************************************C
C                                                                      C
C     Subroutine SECT - Modification of sector.for (May 4, 1986)       C
C                       Omitt subroutines lines and proc. Instead      C
C                       fill arrays ilim and jlim from input.          C
C                                                                      C
C         This subroutine generates a low resolution "sector map."     C
C         This map covers a "wedge shaped" sector of the mapped area,  C
C     which corresponds to a range of azimuth angles from the          C
C     aircraft's position, in which the emitter is "known" to be       C
C     located; thus, this is the only portion of the mapped area       C
C     which will be processed.                                         C
C         This map has points which are spaced coarser than the basic  C
C     maps, specifically, one point for each IRES points, in each      C
C     direction, in the basic maps. Thus the number of points to be    C
C     processed is reduced by a factor of about IRES**2.               C
C         The sector is described by the range of "rows" of the map    C
C     (the "I" coordinates) that encompass the sector, and, for each   C
C     of these rows, the range of "columns" (the "J" coordinate) that  C
C     are in the sector. Since the row and column values are taken in  C
C     steps of IRES, there is the question of "where to start?" The    C
C     low resolution map rows and columns are selected such that the   C
C     aircraft's position falls on one of the low resolution map       C
C     points. (Note that the emitter's position will, in general, not  C
C     fall on a map point.)                                            C
C         Thus, this range of rows, and the corresponding ranges of    C
C     column values, along with IRES, describes the points of the "low C
C     resolution processing sector." (Note that points within the      C
C     "minimum distance," RMIN, are part of the sector and must be     C
C     specially treated in other parts of the program.) A specific     C
C     ordering of these points is set up. The number of points is      C
C     MAXK. Much of the later processing treats these points in this   C
C     serial order, not as points in a two-dimensional grid.           C
C                                                                      C
C----------------------------------------------------------------------C
C                                                                      C
C     Input Files:                                                     C
C                                                                      C
C       * Terminal                                                     C
C                                                                      C
C     Output Files:                                                    C
C                                                                      C
C       * Terminal                                                     C
C       * Unit 2 - Not opened; from calling routines                   C
C                                                                      C
C                                                                      C
C**********************************************************************C
      LOGICAL*4 SECFLG
      INTEGER*2 MAXVAL(MAXSEC),COORDI(MAXSEC),COORDJ(MAXSEC),
     +    MAPCNT(MAXSEC),ILIM(-1:1),JLIM(2,IDN:IUP),ITEMP
      INTEGER*2  ILEFT,IRIGHT,JBELOW,JABOVE
      INTEGER*4 KCHAR(4),J1,J2,IO_R,JO_R,KK,I1,I2,IRES,MAXK,MMAXK,
     +    NCOUNT,K_E,MAXSEC,ERR,IO_C,JO_C,IDN,IUP,JDN,JUP,JLIMIN,JLIMAX
      REAL*4 DISTANCE FROM_E(MAXSEC),GAMMA(0:2),EPSILON,WIDTH,B(2),
     +    SLOPE(2),DELX,DELY,GRAD,ZRES,CUMSUM(MAXSEC),
     +    DISTANCE_FROM_B(MAXSEC),RMIN
C
      PARAMETER (GRAD=57.29577951)
```

```
C
C ******  COMMON blocks:
C
      CHARACTER BLANKL*106,TITLE*106
      INTEGER*4 JMINP,JMAXP,IMINP,IMAXP,JLEN
      COMMON /HIST/ BLANKL,TITLE,JMINP,JMAXP,IMINP,IMAXP,JLEN
C
      INTEGER*4 IO_A,JO_A,IO_A_M,JO_A_M,IO_B,JO_B,IO_B_M,JO_B_M,IO_E,
     +    JO_E,IO_E_M,JO_E_M
      COMMON /POSITS/ IO_A,JO_A,IO_A_M,JO_A_M,IO_B,JO_B,IO_B_M,JO_B_M,
     +    IO_E,JO_E,IO_E_M,JO_E_M
C**********************************************************************C
C**********************************************************************C
C
      ERR=0
C
C ******  Get emitter sector parameters from operator:
C
      WRITE (*,1020)
 1020 FORMAT (' '/'$','Enter LOP sector width (in degrees): ')
      ACCEPT *,WIDTH

      WRITE (2,1000) WIDTH
 1000 FORMAT ('0'/' ','(S) LOP sector width = ',F7.2,' degrees')
C
      WIDTH = WIDTH/GRAD
C
      WRITE (*,1030)
 1030 FORMAT (' '/'$','Enter angle CCW from true LOP to sector edge ',
     +    '(in degrees): ')
      ACCEPT *,EPSILON
C
      WRITE (2,1040) EPSILON
 1040 FORMAT ('0','(S) Angle CCW from true LOP to sector edge = ',
     +    F7.2,' degrees')
C
      EPSILON = EPSILON/GRAD
C
C ******  Calculate sector boundaries lines:
C
      GAMMA(0) = ATAN2 (FLOAT(JO_E-JO_C), FLOAT(IO_E-IO_C))
      GAMMA(2) = GAMMA(0) + EPSILON
      GAMMA(1) = GAMMA(2) - WIDTH
C     CALL LINES(GAMMA,SLOPE,B,JDN,JUP,JO_C,IO_C)
C
C ******  Determine sector boundaries:
C
C     CALL PROC (IO_C,IDN,IUP,JDN,JUP,GAMMA,ILIM,JLIM,SLOPE,B)
C
C Get grid coordinates of center of square sector
C
        type *, ' Enter the grid coordinates of the center of the
     1  square sector (col, row): '
        accept *, icenter, jcenter
C
C Get distance of end points from center of square sector
C
        type *,
     1  ' Enter number of grid points to the right of center: '
        accept *, iright
        type *,
```

```
     1   ' Enter number of ;  d points to the left of center
         accept *, ileft
         type *, ' Enter number of grid points above center: '
         accept *, jabove
         type *, ' Enter number of grid points below center: '
         accept *, jbelow
C
C Fill arrays ilim and jlim
C
         ilim (-1) = max (icenter - ileft, idn)
         ilim (1)  = min (icenter + iright, iup)
C
         do i = ilim (-1), ilim (1)
            jlim (1, i) = max (jdn, jcenter - jbelow)
            jlim (2, i) = min (jup, jcenter + jabove)
         end do
C
C ******  Get desired "low resolution map" resolution from operator:
C
      WRITE (*,1060)
 1060 FORMAT (' '/'$','Enter map resolution (n X 0.2 km):  n = ')
      ACCEPT *,IRES
      WRITE (2,1070) IRES
 1070 FORMAT ('0','(S) Map resolution (n X 0.2 km):  n = ',I2)
C
      ZRES = FLOAT(IRES)
C
C ******  Compute "adjusted I limits" (Rows):
C
      ITEMP = IO_C + IRES*NINT( (ILIM(-1) - IO_C)/ZRES )
      IF (ITEMP.LT.ILIM(-1)) THEN
                                ILIM(-1)=ITEMP+IRES
                              ELSE
                                ILIM(-1)=ITEMP
                              END IF
C
      ITEMP = IO_C + IRES*NINT( (ILIM(1) - IO_C)/ZRES )
      IF (ITEMP.GT.ILIM(1)) THEN
                               ILIM(1)=ITEMP-IRES
                             ELSE
                               ILIM(1)=ITEMP
                             END IF
C
C ******  Compute "adjusted J limits" (Columns):
C
      JLIMIN=JUP
      JLIMAX=JDN
C
      DO 100 IO_R = ILIM(-1), ILIM(1), IRES
      JLIM(1,IO_R) = JO_C + IRES*NINT( (JLIM(1,IO_R) - JO_C) / ZRES )
      IF (JLIM(1,IO_R).LT.JDN) JLIM(1,IO_R) = JLIM(1,IO_R) + IRES
C
      JLIMIN=MIN(JLIMIN,JLIM(1,IO_R))
C
      JLIM(2,IO_R) = JO_C + IRES*NINT( (JLIM(2,IO_R) - JO_C) / ZRES)
      IF (JLIM(2,IO_R).GT.JUP) JLIM(2,IO_R) = JLIM(2,IO_R) - IRES
C
      JLIMAX=MAX(JLIMAX,JLIM(2,IO_R))
  100 CONTINUE
C
C ******  Find the closest "map point" to the true aircraft position:
```

```
C
      IO_A_M = IO_C + IRES*NINT( (IO_A - IO_C)/ZRES )
      IF (IO_A_M.LT.IDN) IO_A_M = IO_A_M + IRES
      IF (IO_A_M.GT.IUP) IO_A_M = IO_A_M - IRES
C
      JO_A_M = JO_C + IRES*NINT( (JO_A - JO_C)/ZRES )
      IF (JO_A_M.LT.JDN) JO_A_M = JO_A_M + IRES
      IF (JO_A_M.GT.JUP) JO_A_M = JO_A_M - IRES
C
      IF (IO_A_M.NE.IO_A.OR.JO_A_M.NE.JO_A) WRITE (2,1085) IO_A_M,JO_A_M
 1085 FORMAT ('0','(S) Coordinates of map point nearest Aircraft are',
     +      ' (',I4,',',I4,')')
C
C ******  Find the closest "map point" to the assumed aircraft position:
C
      IO_B_M = IO_C + IRES*NINT( (IO_B - IO_C)/ZRES )
      IF (IO_B_M.LT.IDN) IO_B_M = IO_B_M + IRES
      IF (IO_B_M.GT.IUP) IO_B_M = IO_B_M - IRES
C
      JO_B_M = JO_C + IRES*NINT( (JO_B - JO_C)/ZRES )
      IF (JO_B_M.LT.JDN) JO_B_M = JO_B_M + IRES
      IF (JO_B_M.GT.JUP) JO_B_M = JO_B_M - IRES
C
      IF (IO_B_M.NE.IO_B.OR.JO_B_M.NE.JO_B) WRITE (2,1090) IO_B_M,JO_B_M
 1090 FORMAT ('0','(S) Coordinates of map point nearest Aircraft'' are',
     +      ' (',I4,',',I4,')')
C
C ******  Find the closest "map point" to the Emitter:
C
      IO_E_M = IO_C + IRES*NINT( (IO_E - IO_C)/ZRES )
      IF (IO_E_M.LT.IDN) IO_E_M = IO_E_M + IRES
      IF (IO_E_M.GT.IUP) IO_E_M = IO_E_M - IRES
C
      JO_E_M = JO_C + IRES*NINT( (JO_E - JO_C)/ZRES )
      IF (JO_E_M.LT.JDN) JO_E_M = JO_E_M + IRES
      IF (JO_E_M.GT.JUP) JO_E_M = JO_E_M - IRES
C
      WRITE (2,1080) IO_E_M, JO_E_M
 1080 FORMAT ('0','(S) Coordinates of map point nearest Emitter are',
     +      ' (',I4,',',I4,')')

C ******  Compute limits for plots:
C
      IF (SECFLG) THEN
C                       IMINP=IO_C+IRES*NINT( (IDN-IO_C)/ZRES )
C                       IF (IMINP.LT.IDN) IMINP = IMINP + IRES
C
C                       IMAXP=IO_C+IRES*NINT( (IUP-IO_C)/ZRES )
C                       IF (IMAXP.GT.IUP) IMAXP = IMAXP - IRES
C
C                       JMINP=JO_C+IRES*NINT( (JDN-JO_C)/ZRES )
C                       IF (JMINP.LT.JDN) JMINP = JMINP + IRES
C
C                       JMAXP=JO_C+IRES*NINT( (JUP-JO_C)/ZRES )
C                       IF (JMAXP.GT.JUP) JMAXP = JMAXP - IRES
C
                      IMINP=ILIM(-1)+IRES*NINT( (IDN-ILIM(-1))/ZRES)
                      IF (IMINP.LT.IDN) IMINP = IMINP + IRES
C
                      IMAXP=ILIM(1)+IRES*NINT( (IUP-ILIM(1))/ZRES )
                      IF (IMAXP.GT.IUP) IMAXP = IMAXP - IRES
```

```
C
                JMINP=JLIMIN+IRES*NINT( (JDN-JLIMIN)/ZRES )
                IF (JMINP.LT.JDN) JMINP = JMINP + IRES
C
                JMAXP=JLIMAX+IRES*NINT( (JUP-JLIMAX)/ZRES )
                IF (JMAXP.GT.JUP) JMAXP = JMAXP - IRES
              ELSE
                IMINP=ILIM(-1)
C                IF (IMINP.GT.IO_A_M) IMINP=IO_A_M
C                IF (IMINP.GT.IO_B_M) IMINP=IO_B_M
                IF (IMINP.GT.IO_E_M) IMINP=IO_E_M
C
                IMAXP=ILIM(1)
C                IF (IMAXP.LT.IO_A_M) IMAXP=IO_A_M
C                IF (IMAXP.LT.IO_B_M) IMAXP=IO_B_M
                IF (IMAXP.LT.IO_E_M) IMAXP=IO_E_M
C
                JMINP=JLIMIN
C                IF (JMINP.GT.JO_A_M) JMINP=JO_A_M
C                IF (JMINP.GT.JO_B_M) JMINP=JO_B_M
                IF (JMINP.GT.JO_E_M) JMINP=JO_E_M
C
                JMAXP=JLIMAX
C                IF (JMAXP.LT.JO_A_M) JMAXP=JO_A_M
C                IF (JMAXP.LT.JO_B_M) JMAXP=JO_B_M
                IF (JMAXP.LT.JO_E_M) JMAXP=JO_E_M
              END IF
C
      JLEN=1+NINT((JMAXP-JMINP)/ZRES)
C
C ******  Locate 4 points around "true" emitter position:
C
      IF (IO_E.EQ.IO_E_M) THEN
                        I1=IO_E_M
                        I2=IO_E_M
      ELSE IF (IO_E.LT.IO_E_M) THEN
                            I1=IO_E_M-IRES
                            I2=IO_E_M
      ELSE
          I1=IO_E_M
          I2=IO_E_M+IRES
      END IF
C
      IF (JO_E.EQ.JO_E_M) THEN
                        J1=JO_E_M
                        J2=JO_E_M
      ELSE IF (JO_E.LT.JO_E_M) THEN
                            J1=JO_E_M-IRES
                            J2=JO_E_M
      ELSE
          J1=JO_E_M
          J2=JO_E_M+IRES
      END IF
C
C ******  Mark emitter position and those around it:
C
      K_E=-1
      DO 220 KK=1,4
      KCHAR(KK)=-1
  220 CONTINUE
C
```

```
      KK = 0
      DO 200 IO_R = ILIM(-1), ILIM(1), IRES
      DO 210 JO_R = JLIM(1,IO_R), JLIM(2,IO_R), IRES
      KK = KK + 1
C
      IF (IO_R.EQ.IO_E_M .and. JO_R.EQ.JO_E_M) K_E = KK
C
      IF (IO_R.EQ.I1      .and. JO_R.EQ.J1)     KCHAR(1) = KK
      IF (IO_R.EQ.I1      .and. JO_R.EQ.J2)     KCHAR(2) = KK
      IF (IO_R.EQ.I2      .and. JO_R.EQ.J1)     KCHAR(3) = KK
      IF (IO_R.EQ.I2      .and. JO_R.EQ.J2)     KCHAR(4) = KK
  210 CONTINUE
  200 CONTINUE
C
C ******  Set elements of arrays that use serial ordering:
C
      KK = 0
      MMAXK=0
      DO 300 IO_R = ILIM(-1), ILIM(1), IRES
      DO 310 JO_R = JLIM(1,IO_R), JLIM(2,IO_R), IRES
C
      KK = KK+1
      IF (KK.GT.MAXSEC) THEN
                          ERR=1
                          GO TO 9999
                        END IF
C
      COORDJ(KK) = JO_R
      COORDI(KK) = IO_R
C
      DELX = IO_R - IO_E
      DELY = JO_R - JO_E
      DISTANCE_FROM_E(KK) = 200.0*SQRT(DELX**2 + DELY**2)
C
      DELX = IO_R - IO_B
      DELY = JO_R - JO_B
      DISTANCE_FROM_B(KK) = 0.2*SQRT(DELX**2 + DELY**2)
C
      MAXVAL(KK) = 0
      CUMSUM(KK) = 0.0

      IF (DISTANCE_FROM_B(KK).LT.RMIN) THEN
                                          MAPCNT(KK)=-1
                                        ELSE
                                          MAPCNT(KK) = 0
                                          MMAXK=MMAXK+1
                                        END IF
  310 CONTINUE
  300 CONTINUE
C
      MAXK = KK
      NCOUNT=MAXK-MMAXK
C
C ******  Return to caller:
C
 9999 RETURN
      END
```

```
EEEEEEEEEE    000000000000(  )000000000000000000000000000000(   )0000    EEEEEEEEEE
EEEEEEEEEE    00000000  Interactive System Design Center  00000000    EEEEEEEEEE
EEEEEEEEEE    0000000000000000000000000000000000000000000000000000    EEEEEEEEEE
```

NOTE: JIM

```
 000    RRRR    SSSS   BBBB    000    SSSS
0   0   R   R   S      B   B   0   0  S
0   0   R   R   S      B   B   0   0  S
0   0   RRRR     SSS   BBBB    0   0   SSS
0   0   R R        S   B   B   0   0      S
0   0   R  R       S   B   B   0   0      S
 000    R   R   SSSS   BBBB    000    SSSS

 SSSS    000    RRRR   TTTTT
S       0   0   R   R    T
S       0   0   R   R    T
 SSS    0   0   RRRR     T
    S   0   0   R R      T
    S   0   0   R  R     T
SSSS     000    R   R    T

FFFFF    000    RRRR     ;;     1     222
F       0   0   R   R    ;;    11    2   2
F       0   0   R   R           1        2
FFFF    0   0   RRRR     ;;     1       2
F       0   0   R R      ;;     1      2
F       0   0   R  R      ;     1     2
F        000    R   R    ;    111   22222
```

File _9DRB5:[ORSBOS.PEL]SORT.FOR;12 (1014,340,0), last revised on 30-NOV-1987 09:26, is a 5 block sequential file owned by UIC [ORSBOS]. The records are variable length with implied (CR) carriage control. The longest record is 72 bytes.

Job SORT (190) queued to SPOCK_LAS2 on 30-NOV-1987 09:27 by user ORSBOS, UIC [ORSBOS], under account A6168 at priority 100, started on printer _SPOCK$TXA4: on 30-NOV-1987 09:36 from queue SPOCK_LAS2.

```
EEEEEEEEEE    0000000000000000000000000000000000000000000000000000    EEEEEEEEEE
EEEEEEEEEE    Digital Equipment Corporation - VAX/VMS Version V4.6    EEEEEEEEEE
EEEEEEEEEE    0000000000000000000000000000000000000000000000000000    EEEEEEEEEE
```

```
      SUBROUTINE SORT(INDE:   COUNT,RANK,MAXK,MMAXK,DISC,MAPC
C*********************************************************************C
C                                                                     C
C     Subroutine SORT - Revision 2/10/86                              C
C                                                                     C
C                                                                     C
C     Input Files: None.                                              C
C                                                                     C
C     Output Files: None.                                             C
C                                                                     C
C     Routines Used: None.                                            C
C                                                                     C
C*********************************************************************C
      INTEGER*4 NCOUNT,ITEMP,MAXK,MMAXK,KK,JJ,IPASS,JJP1
      INTEGER*2 INDEX(10000),RANK(10000),MAPCNT(10000)
      REAL*4 DISC(10000),VAL1,VAL2
C*********************************************************************C
C*********************************************************************C
C
      DO 100 KK = 1,MAXK
      INDEX(KK) = KK
  100 CONTINUE
C
      DO 200 IPASS = 1,MAXK-1
      DO 300 JJ = 1,MAXK-IPASS
      JJP1=JJ+1
C
      VAL1 = DISC(INDEX(JJ))
      VAL2 = DISC(INDEX(JJP1))
C
      IF (VAL1.GT.VAL2) THEN
                            ITEMP = INDEX(JJ)
                            INDEX(JJ) = INDEX(JJP1)
                            INDEX(JJP1) = ITEMP
                          END IF
  300 CONTINUE
  200 CONTINUE
C
      DO 400 JJ = 1,MAXK
      KK = INDEX(JJ)
      RANK(KK) = JJ
  400 CONTINUE
C
C ******  Since the NCOUNT points with MAPCNT's of -1 should have had
C ******  the lowest rankings, "squeeze them out" by reducing all other
C ******  point's rankings by NCOUNT, and set the "deleted" point's
C ******  rankings to zero (which is not otherwise used):
C
      DO KK = 1,MAXK
      IF (MAPCNT(KK).EQ.-1) THEN
                              RANK(KK) = 0
                            ELSE
                              RANK(KK) = RANK(KK) - NCOUNT
                            END IF
      END DO
C
C ******  RETURN TO CALLER:
C
      RETURN
      END
```

Fig.1

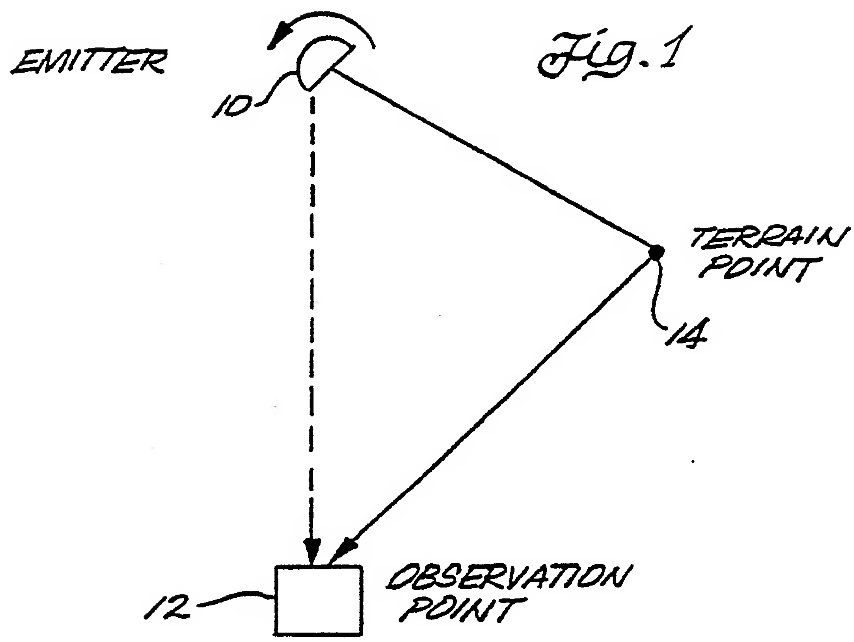


Fig.2

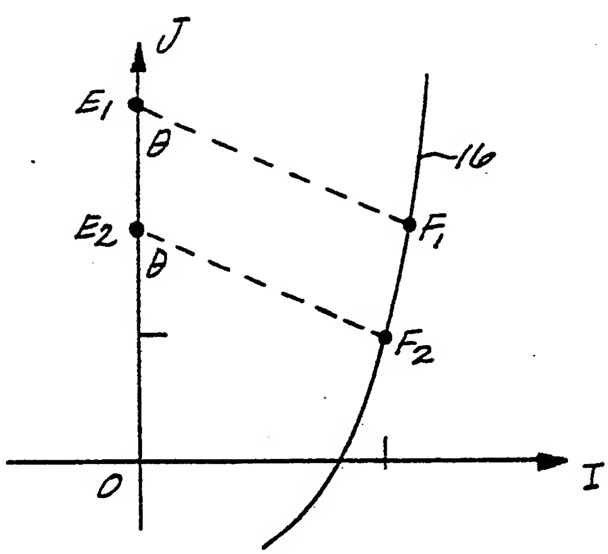
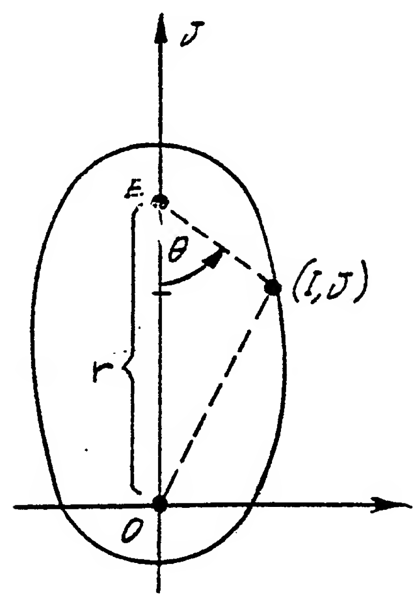


Fig.3



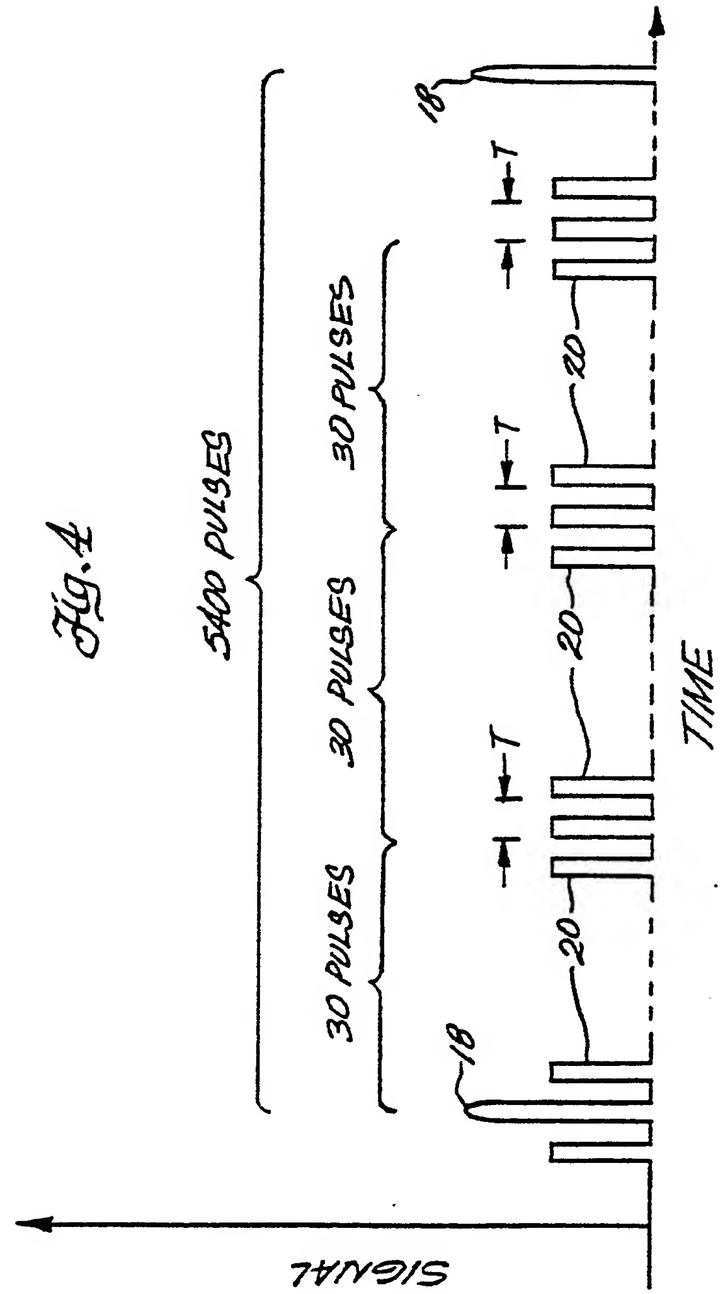
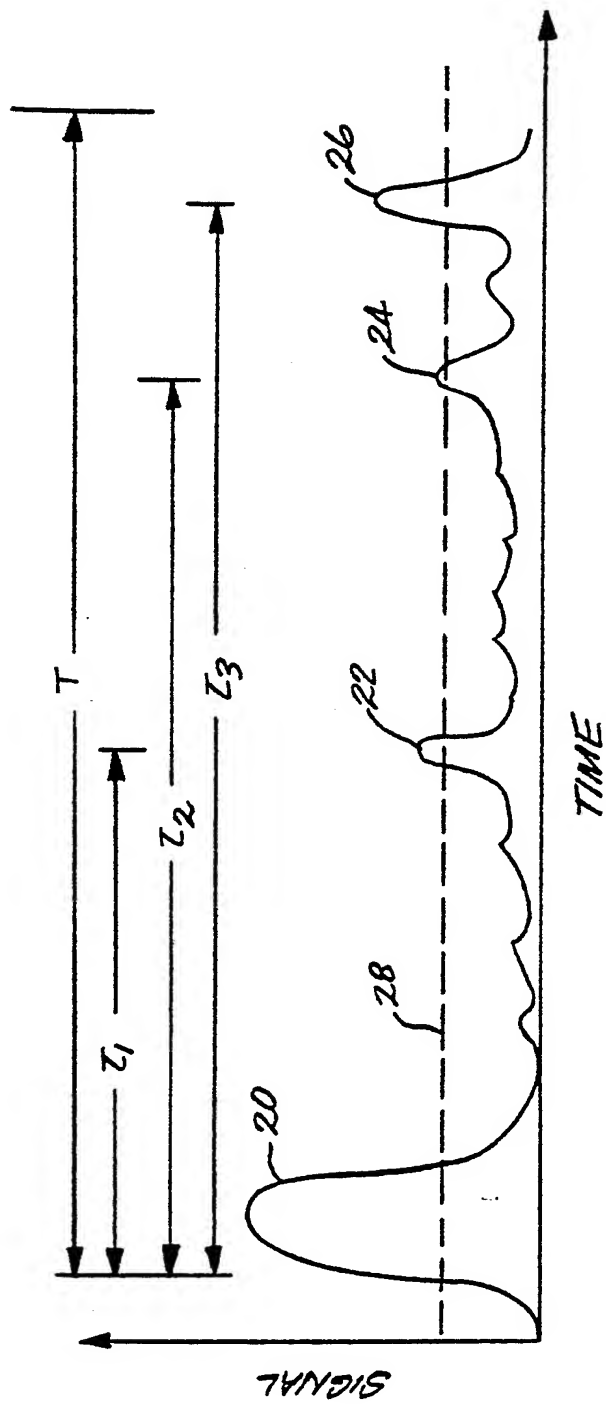

5400 PULSES
30 PULSES
30 PULSES
30 PULSES
18
20
20
20
20
18
T
T
T
TIME
SIGNAL

Fig. 4

Fig 5



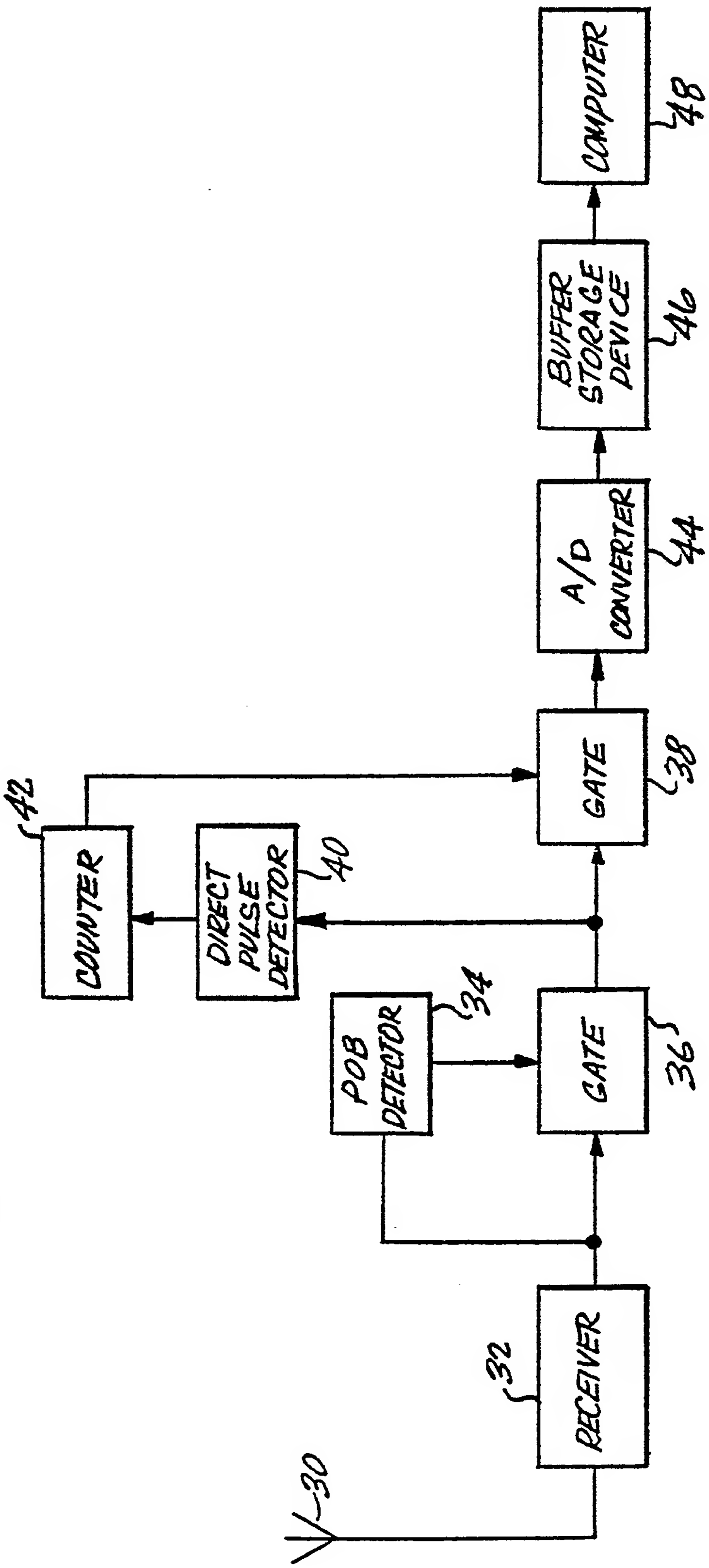

30
RECEIVER
32
POB DETECTOR
34
GATE
36
DIRECT PULSE DETECTOR
40
COUNTER
42
GATE
38
A/D CONVERTER
44
BUFFER STORAGE DEVICE
46
COMPUTER
48

Fig.6

Fig. 7

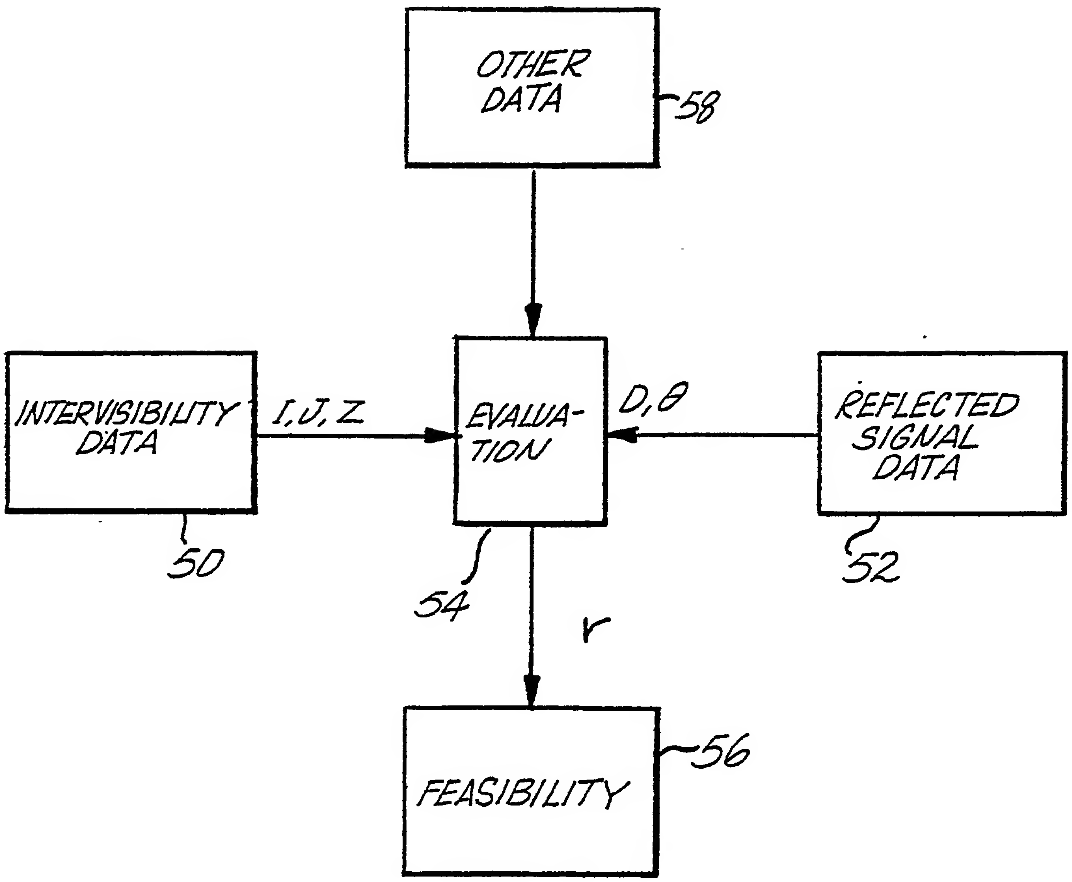

OTHER DATA
58
INTERVISIBILITY DATA
50
I, J, Z
EVALUA-TION
54
D, θ
REFLECTED SIGNAL DATA
52
r
FEASIBILITY
56

Fig. 8A

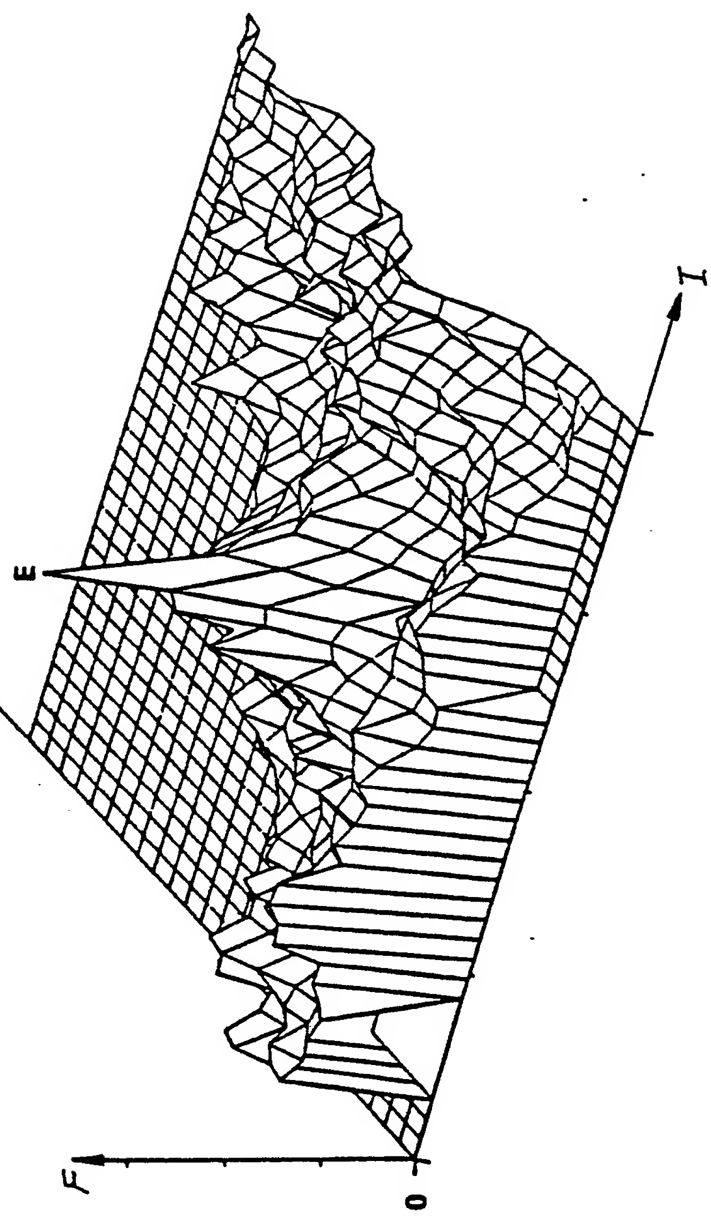


Fig.8B

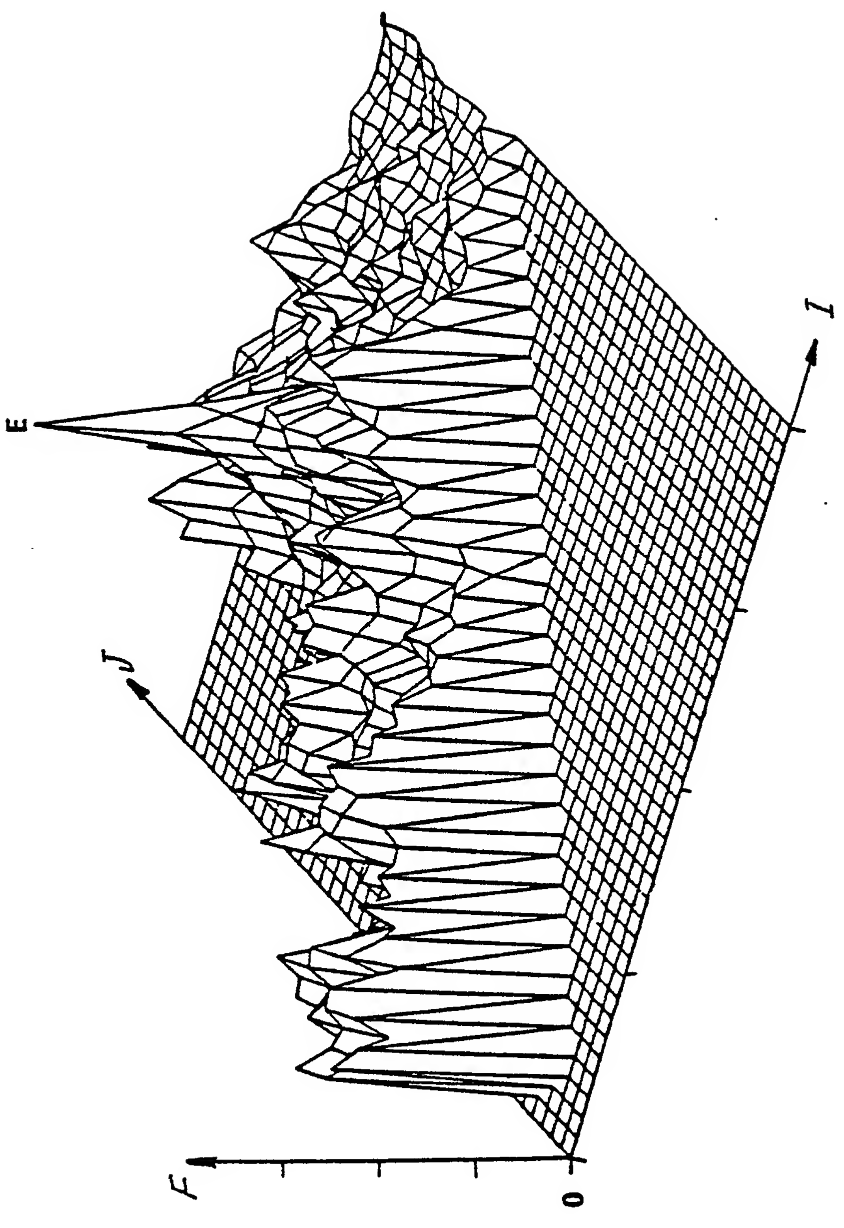


Fig. 8C

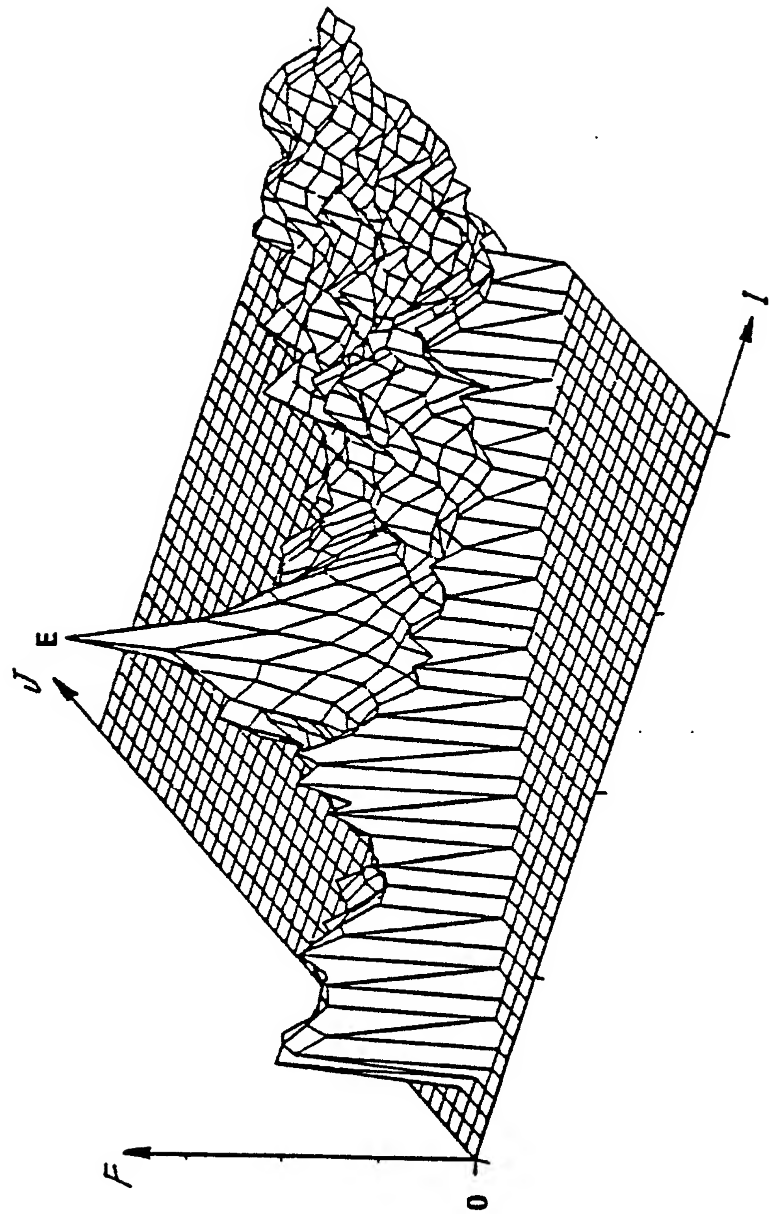


Fig.8D